UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Token Universitário por Meio de Blockchain

Trabalho de Conclusão de Curso

Walan Marcel Teles Oliveira

São Cristóvão – Sergipe

2019

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

Walan Marcel Teles Oliveira

Token Universitário por Meio de Blockchain

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a): Gilton José Ferreira da Silva
Coorientador(a): Hendrik Teixeira Macedo

São Cristóvão – Sergipe

2019

Dedico este trabalho aos meus pais
e ao meu caríssimo irmão.

Agradecimentos

Agradeço primeiramente aos meus pais, Jadisson e Ivanilda, e ao meu irmão, Matheus, por todo o apoio dado. Este trabalho só existe por causa deles.

Agradeço aos amigos e colegas que estiveram presentes nos mais diversos momentos da minha graduação: Alandesson, Anne, Aquino, Ariel, Augusto, Breno, Chefinho, Cleonys, Ézio, Felix, Gabriel, Gabrielle, Iago, Israel, João, Jorge, Juliane, Laís, Laura, Lucas, Luísa, Luiz, Marina, Matheus Barreto, Murilo, Pedro, Raphael, Thaynara, Thiago, Yves, Zé Matheus e tantos outros. Foram muitos rolês, reggaes, trabalhos, cafés, conversas e experiências ao lado de vocês.

E ao caos, que faz tudo se mover.

Tiger got to hunt,
Bird got to fly;
Man got to sit and wonder, "Why, why, why?"

Tiger got to sleep,
Bird got to land;
Man got to tell himself he understand.
(Kurt Vonnegut)

Resumo

Blockchain é uma tecnologia distribuída emergente que oferece um meio confiável e seguro para realização de transações entre participantes sem confiança entre si e que estão dispersos numa rede P2P em larga escala. Como numa forma de incentivo aos alunos universitários continuarem realizando as atividades das matérias em que eles obtiveram boas notas, este trabalho apresenta o desenvolvimento de um sistema de acúmulo dos pontos extras que "sobrarem" nas disciplinas na forma de um *token* utilitário de *blockchain*, o Ludicoin, para poderem ser usados em outras disciplinas. Esse *token* foi implementado numa rede de *blockchain* baseada na plataforma Ethereum. Também foi feita a validação da proposta do produto e suas regras de negócio a partir um questionário usando a escala Likert. A arquitetura do sistema consiste em uma aplicação *web* que, através de um provedor para a API Web3.js, submete transações e chamadas a serem processadas pelos contratos inteligentes contidos na blockchain. Esses contratos inteligentes, confeccionados utilizando a linguagem de programação Solidity, contêm a implementação do *token* Ludicoin e das regras de negócio da aplicação. O projeto faz uso das tecnologias MetaMask, que faz gerenciamento da conta Ethereum do usuário, e Infura, que faz acesso a nós remotos de redes públicas Ethereum. A aplicação *we*b foi implementada em React e se comunica com a *blockchain* utilizando a API Web3.js. Ademais, este trabalho de conclusão de curso apresenta uma revisão bibliográfica acerca dos principais conceitos que envolvem *blockchain* e é realizado um mapeamento sistemático sobre as aplicações dessa tecnologia na área de educação, onde são analisados 22 trabalhos relevantes - somando artigos e patentes - dos 120 encontrados. Este mapeamento apresenta aplicações em diversos setores do sistema educacional, tais como registro de histórico e certificações acadêmicas, método de recompensar os alunos, forma de compartilhamento de material didático garantindo os direitos autorais, sistema de aprendizado personalizado, entre outros. Por fim, o sistema produzido neste trabalho foi bem-sucedido em realizar as funcionalidades necessárias para o cumprimento da proposta, utilizando a rede Ethereum pública de teste Rinkeby. Os resultados do questionário indicam que há grande aceitação da proposta do projeto, porém as regras de negócio ainda precisam ser refinadas de modo que sejam satisfatórias para alunos e professores, e possam causar um impacto maior no desempenho dos estudantes. Como trabalhos futuros, são sugeridas a aplicação do sistema em turmas universitárias, a implantação do sistema em uma rede de *blockchain* privada e a implementação de novas regras para concessão de Ludicoins.

Palavras-chave: *Blockchain*, Criptomoeda, Contratos Inteligentes, *Tokens*, Educação.

Abstract

Blockchain is an emerging distributed technology that provides a reliable and secure means for conducting transactions between participants who do not trust each other and dispersed on a large-scale P2P network. As a way to encourage university students to keep performing the activities of the courses in which they obtained good grades, this work shows the development of a system of accumulation of the extra credits that remained unused in the courses, in the form of a blockchain utility token, the Ludicoin, to be used in other courses. This token was implemented in a blockchain network based on the Ethereum platform. Plus, the product proposition and its business rules were validated through a questionnaire using the Likert scale. The system architecture consists of a web application that, through a provider for the Web3.js API, submits transactions and calls to be processed by the smart contracts contained in the blockchain. These smart contracts, which are built on Solidity programming language, contain the implementation of Ludicoin token and the business rules of the application. The project makes use of the following tools: MetaMask, which manages the user's Ethereum account, and Infura, which accesses remote nodes on Ethereum public networks. The web application was implemented in React and communicates with the blockchain by using the Web3.js API. This bachelor's thesis also presents a bibliographical review about the main concepts regarding blockchain and a systematic mapping is carried out regarding the applications of this technology in the area of education, where 22 relevant works of the 120 works found - counting articles and patents - are analyzed. This mapping presents applications in various sectors of the educational system, such as registering academic records and certifications, methods of rewarding students, teaching material sharing that ensures copyright, personalized learning system, among others. Finally, the system produced in this work succeeded in delivering the functionalities necessary for the fulfillment of the proposal, using the Rinkeby public test Ethereum network. The results of the questionnaire indicate that there is a wide acceptance of the project proposal, but its business rules still need to be refined so that they are satisfactory to students and teachers, and may have a greater impact on student performance. Recommended future work includes the application of the system in university classes, the deployment of the system in a private blockchain network and the implementation of new rules for granting Ludicoins.

Keywords: Blockchain, Cryptocurrency, Smart Contracts, Tokens, Education.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
CRUD	Create - Read - Update - Delete
HTTP	HyperText Transfer Protocol
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IoT	Internet of Things
JSON	Javascript Object Notation
MVP	Minimum Viable Product
P2P	Peer-to-peer

Sumário

1 Introdução

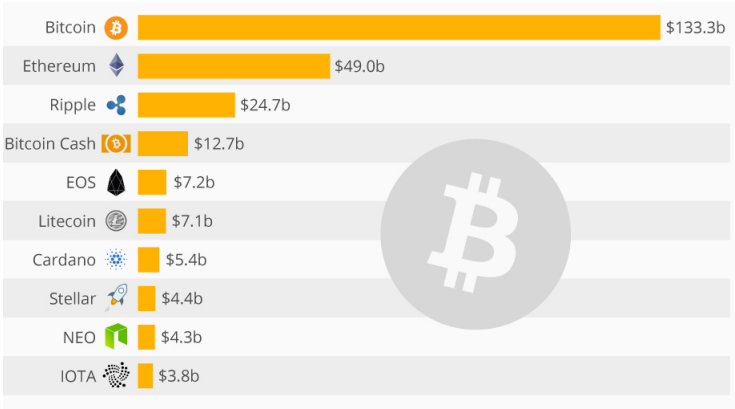
A tecnologia *blockchain* vem despertando crescente interesse desde a sua concepção em 2008 com a publicação do artigo de Satoshi Nakamoto sobre Bitcoin (NAKAMOTO, 2008). Várias aplicações dessa tecnologia vêm sendo exploradas no mercado desde então e, mais recentemente, ela está sendo tema de um número cada vez maior de pesquisas e projetos científicos (TAYLOR et al., 2019).

A ideia básica de um sistema de *blockchain* é funcionar como um livro-razão público imutável e distribuído, onde os registros são armazenados em blocos que são replicados entre os participantes da rede, permitindo serem verificados e auditados.

Uma de suas principais promessas é a de garantir confiabilidade em um ambiente sem uma entidade central de confiança. Esta confiabilidade é obtida a partir de um consenso distribuído na rede que usa funções hash para criptografia (GRECH; CAMILLERI, 2017). Tal consenso distribuído proporciona um ambiente seguro e escalável para o armazenamento de transações. Essa é a base das várias moedas baseadas em *blockchain*, as chamadas "criptomoedas", que operam sem uma autoridade central. Atualmente, as criptomoedas são a principal aplicação de dessa tecnologia e formam um mercado avaliado em bilhões de dólares. No gráfico da figura 1, datado de 13 de abril de 2018, tem-se o valor total de mercado das principais criptomoedas.

Em 2013, tem início a fase chamada *Blockchain* 2.0 (BASHIR, 2017), onde foram possibilitadas duas importantes funcionalidades em *blockchains*: os "contratos inteligentes" (do inglês, *smart contracts*) e a criação de *tokens* digitais. "Contratos inteligentes" são *scripts* contendo alguma lógica de negócio que operam sobre a *blockchain*, e são executados e armazenados como transações na cadeia de blocos (Christidis; Devetsikiotis, 2016). *Tokens* são ativos digitais que possuem alguma função dentro do projeto no qual estão inseridos, seja para funcionar como ações de uma empresa, método de pagamento interno do ecossistema do projeto ou dar acesso à estrutura funcional do projeto. *Tokens* servem para possibilitar funcionalidades mais amplas que as de uma moeda. Essas duas tendências permitem que regras de negócios e organizações

Figura 1 – Valor de mercado das 10 principais criptomoedas.



Fonte: https://www.statista.com/chart/13520/the-top-ten-cryptocurrencies/

possam ser executadas de forma autônoma, expandindo o campo de ação das cadeias de blocos para além de transações financeiras simples.

A tecnologia *blockchain* tem um potencial de transformação imenso e diversas aplicações já estão surgindo em vários setores: artes, saúde, finanças, governo, entre outros. Na própria área da computação temos aplicações em protocolos de redes, nuvem, névoa, IoT, etc.

Dentro do contexto acadêmico, foi identificada pelo autor a falta de motivação dos alunos em realizarem todas as atividades em disciplinas que eles possuem pontuação elevada. Assim, foi formulada a hipótese de que guardar pontos extras para usar em outras unidades ou disciplinas pode ser útil para motivar os alunos a realizarem todas as suas atividades. A partir disso foi pensado um sistema de acúmulo dos pontos sobressalentes nas disciplinas curriculares para que possam ser utilizados posteriormente, como uma forma de incentivo para que o aluno continue realizando as atividades das matérias em que ele vai bem. Assim, quando um aluno tiver atingido a nota máxima e tiver feito alguma atividade valendo ponto extra, ele poderá utilizar esses pontos em uma unidade posterior ou em outra disciplina. Além disso, esses pontos podem ser representados em forma de moedas como um elemento de ludificação.

Busca-se que esse sistema seja confiável, para gerar aceitação entre os professores; que seja possível fazer o acompanhamento da origem dos pontos pelas partes interessadas e que possa funcionar sem a necessidade de que o sistema seja mantido por uma entidade intermediária central, o que acarretaria em custos operacionais mais elevados. Além disso, é desejável que o sistema seja facilmente escalável. Tais requisitos se alinham com as principais características técnicas alcançáveis por meio de *blockchain* e, por conseguinte, esta foi a técnica escolhida para a realização de tal sistema neste trabalho.

Para a validação do sistema foi escolhida a aplicação de questionário com respostas em escala Likert, onde os participantes respondem especificando seu nível de concordância com cada afirmação apresentada (LIKERT, 1932).

1.1 Objetivos

Nesta seção estão descritos o objetivo geral e os objetivos específicos com o intuito de elucidar o escopo e o direcionamento do presente trabalho.

1.1.1 Objetivo Geral

O presente trabalho tem como objetivo desenvolver um sistema de acúmulo de pontos sobressalentes nas disciplinas da universidade a partir da criação de um *token* de *blockchain* para ser usado por alunos e professores do ambiente acadêmico. Esse sistema possibilitará ao aluno guardar pontos extras que sobrarem em unidades das disciplinas e utilizá-los em unidades posteriores ou em outras disciplinas.

1.1.2 Objetivos Específicos

- Realizar o mapeamento sistemático das pesquisas e projetos de *blockchain* para a educação;
- Desenvolver um *token* de *blockchain* para acúmulo de pontos extras de estudantes universitários nas disciplinas;
- Validar o produto gerado com um grupo de teste baseado na escala Likert.

1.2 Estrutura do Documento

O trabalho está organizado da seguinte forma:

- Capítulo 1 - Introdução. Neste capítulo estão apresentados o contexto, as motivações e os objetivos do trabalho;
- Capítulo 2 - Fundamentação Teórica. Neste capítulo estão apresentados os conceitos relevantes para o entendimento do projeto;
- Capítulo 3 - Materiais e Métodos. Neste capítulo são apresentadas as tecnologias utilizadas, o método seguido, o modelo do sistema e o desenvolvimento.
- Capítulo 4 - Trabalhos Relacionados. Neste capítulo é apresentado um mapeamento sistemático das pesquisas e patentes existentes relacionadas ao tema;
- Capítulo 5 - Resultados e Discussão. Neste capítulo são apresentados e discutidos os resultados do trabalho;
- Capítulo 6 - Conclusão e Trabalhos Futuros. Neste capítulo são apresentadas as conclusões do trabalho e sugestões de trabalhos futuros;

Ao fim do documento são apresentados as referências e os apêndices.

2

Fundamentação Teórica

Este capítulo apresenta os principais conceitos e fundamentos envolvidos no sistema proposto, no intuito de prover maior embasamento teórico ao que será explanado posteriormente. São aqui descritos os conceitos de *Blockchain*, Criptomoedas, Contratos Inteligentes e *Tokens* de *Blockchain*.

2.1 Definição de Blockchain

Uma cadeia de blocos criptograficamente protegida foi descrita pela primeira vez no artigo *How to time-stamp a digital document* de Haber e Stornetta (1991). O objetivo deles era implementar um sistema onde as estampas de tempo (*timestamps*) de documentos não pudessem ser adulteradas. No ano seguinte, eles adicionaram árvores Merkle ao design do projeto, o que aumentou sua eficiência ao permitir que vários certificados para documentos fossem coletados em um bloco.

Tomando como base esse projeto, foi publicado em 2008 o artigo *Bitcoin: A Peer-to-Peer Electronic Cash System* (NAKAMOTO, 2008) assinado sob o pseudônimo de Satoshi Nakamoto. Neste artigo é descrito o que se convencionou chamar de *blockchain*. As palavras *block* (bloco) e *chain* (cadeia) foram usadas separadamente no artigo original de Nakamoto, mas se popularizaram como uma única palavra.

Há diversas definições de *blockchain*, dependendo da perspectiva adotada. Bashir (2017) apresenta três definições técnicas:

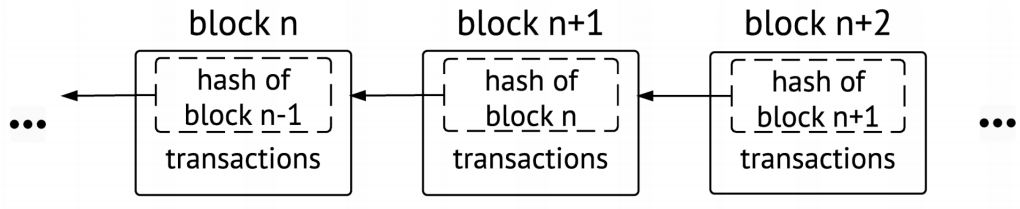
- Em uma *blockchain*, todos os pares eventualmente chegam a um acordo em relação ao estado de uma transação;
- *Blockchain* é um livro-razão distribuído e compartilhado. *Blockchain* pode ser considerado um *ledger* (livro-razão) de transações. As transações são ordenadas e agrupadas em blocos. Atualmente, o principal modelo usado no mundo real é baseado em bancos de dados privados mantidos por suas

respectivas organizações, ao passo que um livro-razão distribuído pode servir como uma fonte única da verdade para todas as organizações que estão usando a *blockchain*;

- *Blockchain* é uma estrutura de dados; Ela é basicamente uma lista encadeada que usa ponteiros *hash* ao invés de ponteiros normais. (p. 18)

De maneira geral, pode-se definir *blockchain* como um *ledger* (livro-razão) distribuído, *peer-to-peer* protegido criptograficamente, *append-only* (estritamente incremental), imutável, e atualizável apenas via consenso ou acordo entre os pares (BASHIR, 2017). A figura 2 destaca a visualização da cadeia de blocos como uma lista encadeada de ponteiros *hash*.

Figura 2 – Ilustração de uma blockchain e seus ponteiros *hash*.



Fonte: (Christidis; Devetsikiotis, 2016).

2.2 Propriedades de uma Blockchain

Greve et al. (2018) elenca as seguintes propriedades da *blockchain* como as principais inovações trazidas por esta tecnologia no desenvolvimento de sistemas:

- Descentralização: o controle e autoridade é distribuído entre os nós participantes da *blockchain*, onde um mecanismo de consenso é usado para se chegar a um acordo quanto à validade das transações, sem a necessidade de uma entidade intermediária confiável. Este é um conceito essencial quando se fala em *blockchain*;
- Disponibilidade e integridade: o sistema é baseado em milhares de nós em uma rede *peer-to-peer* e os dados são replicados e atualizados em cada nó de maneira segura, dessa forma o sistema se mantém disponível e consistente;
- Transparência e auditabilidade: o *ledger* é publicamente compartilhado e todo mundo pode ver as transações registradas, isto permite o sistema ser transparente e auditável;
- Imutabilidade: uma vez que um dado foi registrado na *blockchain*, é extremamente difícil - quase impossível - de modificá-lo;
- Privacidade e anonimidade: não há terceiros envolvidos com acesso e controle sobre os dados dos usuários. O mecanismo de assinatura digital a partir de um sistema criptográfico com chaves públicas e privadas, permite que as transações sejam até certo ponto anônimas;

- Desintermediação: essa tecnologia permite a integração de vários sistemas de forma direta e eficiente, permitindo a eliminação de intermediários e simplificando o projeto dos sistemas e processos;
- Cooperação e incentivos: são estabelecidos incentivos, baseados na teoria dos jogos, para a cooperação entre os participantes da rede.

2.3 Elementos de uma Blockchain

Nesta seção serão apresentados os principais elementos que compõem uma *blockchain*.

2.3.1 Funções *Hash* Criptográficas

Uma função *hash* criptográfica h: $\{0,1\}^* \rightarrow \{0,1\}^n$ mapeia o conjunto $\{0,1\}^*$ de todas as *strings* binárias para o conjunto $\{0,1\}^n$ de todas as *strings* de comprimento fixo n. Além disso, espera-se que essas funções sejam resistentes à colisão, ou seja, deve ser extremamente improvável encontrar quaisquer duas cadeias $x \neq x'$ tais que h(x) = h(x') (HOFFSTEIN et al., 2008). Essas funções ajudam a verificar a integridade dos dados. Assim, dado um dado p, podemos calcular seu *hash* y = h(p) e checar se p não foi modificado desde então recomputando h(p) e verificando se é igual a y. Exemplos de funções *hash* criptográficas que são usadas em *blockchains* Bitcoin e Ethereum são SHA-256 e Keccak-256, respectivamente. Uma avaliação h(y) de alguma entrada y é referida como "o *hash* de y".

2.3.2 Assinaturas Digitais

As funções *hash* criptográficas têm propriedades que as tornam adequadas para uso como meio de verificar a integridade de uma mensagem e como parte de um método de assinatura digital (PENARD; WERKHOVEN, 2008). Este método consiste em uma chave privada *sk*, uma chave pública *pk* e as funções Assinar (M, *sk*), que produz uma assinatura S, e Verificar (M, S, pk), que retorna um booleano indicando se o dado S é um assinatura válida para a mensagem M. Essa assinatura deve provar a autenticidade e integridade da mensagem, de tal forma que podemos ter certeza de que realmente foi o remetente da mensagem que a enviou, e que realmente foi essa mensagem que ele enviou. Finalmente, podemos usar a assinatura para provar que o remetente a enviou e que ninguém mais poderia ter feito isso. Em sistemas de *blockchain*, todos os usuários possuem uma chave pública, que serve como um identificador, e uma chave privada para o uso desse método.

2.3.3 Rede *Peer-to-Peer*

Por definição, *blockchain* é composto por uma rede *peer-to-peer* (P2P), na qual cada dispositivo participante atua como um nó na rede. *Peer-to-peer* é um modelo de comunicação

não hierárquica onde os indivíduos podem se comunicar livremente pela rede sem uma divisão fixa entre cliente e servidor (TANENBAUM, 2003). Os pares (*peers*) são parceiros na rede com iguais privilégios e influência no ambiente.

2.3.4 Sistema Distribuído

Blockchain é na sua essência um sistema distribuído. Um sistema distribuído é uma "coleção de computadores independentes entre si que se apresenta ao usuário como um sistema único e coerente" (TANENBAUM; STEEN, 2006, p.2). Outra definição bem estabelecida o descreve como uma "coleção de computadores autônomos interligados através de uma rede de computadores e equipados com software que permita o compartilhamento dos recursos do sistema: hardware, software e dados" (COULOURIS et al., 2011, p.2).

Pode-se chamar um participante num sistema distribuído de nó. Nós são capazes de enviar e receber mensagens um do outro. Os nós podem ser honestos, defeituosos ou maliciosos. Um nó que pode exibir um comportamento arbitrário também é conhecido como um nó bizantino. Esse comportamento arbitrário pode ser intencionalmente malicioso, o que é prejudicial para a operação da rede (BASHIR, 2017).

Os principais desafios no projeto de sistemas distribuídos são a coordenação entre os nós e a tolerância a falhas (BASHIR, 2017). Mesmo que alguns dos nós tornem-se defeituosos ou que os links de rede quebrem, o sistema distribuído deve tolerar isso e deve continuar a trabalhar sem falhas para alcançar o resultado desejado. Os elementos a seguir descrevem mecanismos para solucionar esses problemas.

2.3.5 Blocos

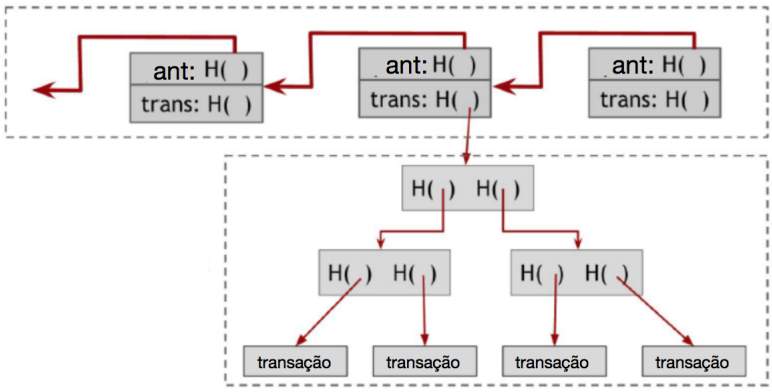
A tecnologia *blockchain* é formada por blocos encadeados por ponteiros *hashes* numa lista. Ponteiro *hash (hash pointer)* é um apontador para onde se encontra armazenado o dado d e o seu *hash* criptográfico H(d). Dessa forma, ele torna possível a recuperação do dado d e a sua verificação, a partir de H(d).

Denomina-se de bloco *genesis* o primeiro bloco da lista. Cada bloco contém um conjunto de transações. Elas são estruturados numa estrutura de árvore binária de ponteiros hash, geralmente numa estrutura denominada árvore Merkle (ou alguma variação dela). A Figura 3 destaca essa estrutura.

Greve et al. (2018, p. 13) traz uma explicação clara e concisa deste tipo de estrutura de dados dados, descrevendo-a como segue:

> Nesta árvore, no último nível, estão as folhas que contêm os dados (especificamente, apontadores para as transações propriamente ditas); no penúltimo nível, os pais possuem apontadores *hash* para esses dados. Em seguida, em cada nível, os pais apontadores são agrupados dois a dois, até que se alcance a raiz

Figura 3 – Blockchain com árvore binária Merkle.



Fonte: (NARAYANAN et al., 2016) (Adaptado).

> da árvore. O *hash* da raiz da árvore é então armazenado de forma segura dentro do *header* do bloco, em conjunto com demais informações. Pela propriedade dos apontadores *hash*, se houver qualquer modificação nos dados da árvore, ela será identificada pela verificação dos *hashes*. Além disso, o espaço necessário para o seu armazenamento é muito pequeno, comparativamente ao tamanho dos dados (p. 13).

O estudo de Greve et al. (2018, p. 13) ainda aponta que:

> Uma outra vantagem de uso da árvore de Merkle é a facilidade de prova de filiação (*proof of membership*), sendo possível verificar se uma transação t_i pertence a árvore T em tempo *O(log(n))*, onde n é o conjunto de nós da árvore. Assim, para provar que $t_i \in$ T, basta apresentar os *hashes* do caminho em que t_i é folha seguindo até a raiz da árvore. Esse caminho de autenticação ou caminho de Merkle terá o *hash* do complemento dos pares, a cada nível (p.13).

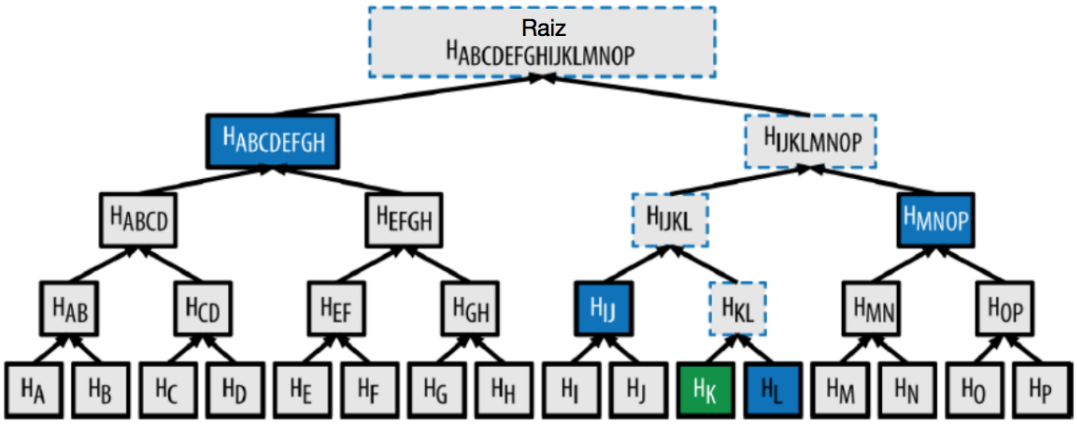
A complexidade algorítmica para um nó da rede calcular os *hashes* complementares a partir do caminho Merkle é de *O(log(n))*. A título de ilustração, na Figura 4, o caminho de merkle para se verificar que a transação H_K pertence à árvore passa por quatro hashes: H_L, H_{IJ}, H_{MNOP}, $H_{ABCDEFG}$.

Os blocos principais campos formadores de um bloco são: *header* (cabeçalho); ponteiro *hash* para o bloco anterior; ponteiro *hash* para a árvore Merkle; *nonce*, caso o consenso seja por *Proof-of-Work*; e o *hash* do bloco, que usa todos os outros dados do bloco como entrada da função de *hash*. No *header* é que se encontra o *timestamp* do bloco. Os campos podem variar de acordo com a implementação da *blockchain*.

2.4 Consenso

Consenso é um processo de acordo entre nós sem confiança entre si sobre o estado final dos dados. Para atingir o consenso, diferentes algoritmos podem ser usados. É fácil alcançar um

Figura 4 – Caminho de Merkle.



Fonte: (ANTONOPOULOS, 2017) (Adaptado).

acordo entre dois nós (por exemplo, em sistemas cliente-servidor) mas quando múltiplos nós estão participando em um sistema distribuído e eles precisam concordar sobre um certo valor, se torna muito difícil atingir um consenso. Este conceito de alcançar um consenso entre múltiplos nós é conhecido como consenso distribuído (BASHIR, 2017). Nesta seção são apresentados os principais problemas e protocolos de consenso distribuído.

2.4.1 Problema do Gasto Duplo

Chohan (2017, p. 2) traz a seguinte definição sobre o problema do gasto duplo:

> O problema do gasto duplo é um risco em potencial para criptomoedas, ou qualquer outro esquema de dinheiro digital, onde uma mesma moeda pode ser gasta mais de uma vez, e isto é possível porque uma moeda digital consiste em um arquivo digital que pode ser duplicado ou falsificado (p. 2).

Ele pode ocorrer quando, por exemplo, um usuário envia moedas para dois usuários diferentes ao mesmo tempo e essas transações são verificadas de maneira independente uma da outra. O criador do Bitcoin, Satoshi Nakamoto, estava consciente deste problema e o incluiu no artigo seminal que descreveu o Bitcoin (NAKAMOTO, 2008). Assim sendo, uma das funções do consenso é evitar que este tipo de falha ocorra.

2.4.2 Problema dos Generais Bizantinos

Em 1982, um experimento mental foi proposto por Lamport, Shostak e Pease (1982) onde um grupo de generais que estão liderando diferentes partes do exército bizantino estão planejando atacar ou recuar de uma cidade. A única forma de comunicação entre eles é um mensageiro e eles precisam concordar em atacar ao mesmo tempo para vencer. O problema é que um ou mais generais podem ser traidores e podem comunicar uma mensagem enganosa. Portanto,

existe uma necessidade de encontrar um mecanismo viável que permite o acordo entre generais mesmo na presença de generais traiçoeiros para que o ataque ainda possa ocorrer ao mesmo tempo. Fazendo uma analogia com sistemas distribuídos, generais podem ser considerados nós, traidores podem ser considerados nós maliciosos, e o mensageiro pode ser entendido como um canal de comunicação entre os generais (BASHIR, 2017).

Este problema foi resolvido por Castro e Liskov (1999), que apresentaram o algoritmo de *Practical Byzantine Fault Tolerance* (PBFT). Mais tarde, em 2009, a primeira implementação prática foi feita com a invenção do Bitcoin, onde o algoritmo de *Proof-of-Work* (PoW) foi desenvolvido como um mecanismo para alcançar consenso. Desde então outros protocolos de consenso foram criados para resolver este mesmo problema, garantindo sistemas com *Byzantine Fault Tolerance* (Tolerância à Falha Bizantina), ou seja, sistemas capazes de chegar a um acordo correto mesmo na presença de nós maliciosos.

2.4.3 Protocolos de Consenso

O protocolo de consenso *Proof-of-Work*, pelo seu pioneirismo, lançou as bases para todos os outros protocolos. Processos como, por exemplo, a substituição da cadeia menor pela cadeia mais longa, são aproveitados pelos outros protocolos. Antes da validação, as transações submetidas ficam em uma lista criptograficamente protegida de transações não validadas mantida e atualizada pelos nós da rede de blockchain.

2.4.3.1 *Proof-of-Work* (Prova de Trabalho)

Este é o algoritmo de consenso mais usado, sendo inaugurado no lançamento do Bitcoin. Ele é uma prova de que recursos computacionais suficientes foram gastos para se produzir um bloco válido. Neste modelo, os nós, chamados também de mineradores, competem para resolver um problema criptográfico. Este problema é encontrar um valor que, quando concatenado com os outros dados do bloco (transações, *timestamps*, *hash* do bloco anterior, etc) na entrada de uma função, *hash* gera uma saída menor que um valor alvo estipulado pelo sistema (BASHIR, 2017). Este valor é procurado a partir de um campo do bloco chamado *nonce*, que vai sendo incrementado até encontrar um valor que satisfaça essa condição. Quando a condição é satisfeita, um novo bloco é minerado e o nó que o minerou é recompensado. Transações são selecionadas, processadas e, enfim, armazenadas nesse novo bloco. Em seguida, o novo bloco é transmitido para todos os nós da *blockchain*.

O esforço computacional resolver o problema é exponencial em relação ao número de zeros à esquerda do alvo e pode ser verificado com uma única execução da função *hash* (NAKAMOTO, 2008). No caso do Bitcoin, a função *hash* é a SHA-256. Para se obter uma taxa constante de geração de blocos, o valor do alvo pode ser aumentado ou diminuído de acordo com o número de nós, tornando-se mais difícil quando há muitos mineradores e mais fácil quando há poucos.

Caso ocorra de haverem duas cadeias de blocos diferentes, a menor cadeia será substituída pela maior (NAKAMOTO, 2008). Isso pode acontecer quando dois nós mineram um mesmo bloco com pouca diferença de tempo entre os dois; essas duas cadeias permaneceram em paralelo até que uma fique maior que a outra. Note que, como a criação de um valor de *hash* válido envolve os dados do bloco e o *hash* do bloco anterior, para modificar um bloco no meio da cadeia, um atacante teria que refazer o Proof-of-Work do bloco e de todos os blocos depois dele e ainda alcançar ou ultrapassar a cadeia com blocos minerada por nós honestos.

2.4.3.2 *Proof-of-Authority* (Prova de Autoridade)

Nas redes baseadas em *Proof-of-Authority*, transações e blocos são validados apenas por nós aprovados, conhecidos como validadores. Nesse processo, um nó validador é escolhido aleatoriamente para gerar um novo bloco; ele coleta as transações que farão parte do bloco e as executa. Transações que são inválidas ou que não podem ser executadas são rejeitadas, como, por exemplo, uma transação que tenta gastar mais moedas do que o usuário tem. Em seguida, ele assina assina o bloco com a sua chave privada e envia o bloco para os outros validadores. Para validar o novo bloco, os outros validadores checam a assinatura do criador do bloco e executam as transações para verificarem se elas são válidas, ou seja, se elas são executadas com sucesso. Se a validação do bloco é bem-sucedida, ele é adicionado à *blockchain*; caso contrário, ele é rejeitado e a reputação do criador do bloco é afetada negativamente.

É necessário garantir que os nós participantes nesse processo de consenso não sejam maliciosos. Com o *Proof-of-Authority*, os indivíduos ganham o direito de se tornarem validadores; portanto, há um incentivo para manter a posição que conquistaram. Ao associar uma reputação à identidade, os validadores são incentivados a manter o processo de transação, pois não desejam ter suas identidades associadas a uma reputação negativa - correndo o risco de serem banidos.

2.4.3.3 *Proof-of-Stake* (Prova de Participação)

Em um sistema de *Proof-of-Stake* (PoS), um validador de transação (isto é, um nó na rede) deve provar que possui um certo ativo para participar da validação das transações (HOUBEN; SNYERS, 2018). No caso das criptomoedas, esse ativo é a quantidade de moedas, então, dependendo de quantas moedas o participante possui, ele terá uma chance maior ou menor de ser aquele que irá validar o próximo bloco. Cada nó tem uma chance proporcional à sua participação monetária no sistema. Dessa forma, um nó com 300 moedas terá 3 vezes mais chances de ser escolhido que um nó com 100 moedas. As partes envolvidas na transação pagam uma taxa ao validador da transação pelos seus serviços de validação. Existem ainda algumas variações do PoS, onde o critério é a idade das moedas dos nós: quanto mais antigas, maiores as chances de ser escolhido para a validação. Criptomoedas como Neo e Ada (da plataforma Cardano) utilizam PoS como mecanismo de consenso.

2.4.3.4 *Delegated-Proof-of-Stake* (Prova de Participação Delegada)

É uma variação do *Proof-of-Stake* onde cada nó que possui ativos no sistema pode delegar a validação de uma transação para outros nós por meio do votação (BASHIR, 2017). Ou seja, neste algoritmo os participantes não decidem diretamente se um bloco é válido ou não, eles votam de "delegados"que, por sua vez, fazem a validação.

2.4.3.5 Proof-of-Activity (Prova de Atividade)

Este é um protocolo híbrido que combina *Proof-of-Work* com *Proof-of-Stake* (BENTOV et al., 2014). Em uma *Proof-of-Activity*, a mineração começa com *Proof-of-Work*, com mineradores disputando para resolver um quebra-cabeça criptográfico. Dependendo da implementação, os blocos extraídos não contêm nenhuma transação (funcionando como um *template*), portanto, o bloco vencedor conterá apenas um cabeçalho e o endereço de recompensa do minerador.

A partir disso, o sistema muda para *Proof-of-Stake*. Com base nas informações do cabeçalho, um grupo aleatório de validadores é escolhido para assinar o novo bloco. Quanto mais moedas no sistema um validador possui, maior a probabilidade de ele ser escolhido. O novo bloco se torna um bloco completo assim que todos os validadores o assinam. Se alguns dos validadores selecionados não estiverem disponíveis para concluir a validação do bloco, o próximo bloco vencedor será selecionado, um novo grupo de validadores será escolhido e assim por diante, até que um bloco receba a quantidade correta de assinaturas. As taxas são divididas entre o minerador e os validadores que assinaram o bloco.

2.5 Tipos de Blockchain

Existem múltiplos tipos de *blockchain* dependendo do modelo de negócio no qual a tecnologia é aplicada. Muitas vezes as definições não são claras ou até mesmo redundantes. Por isso serão descritos nessa subseção apenas os três tipos mais abrangentes.

2.5.0.1 Pública

Essas *blockchains* estão abertas ao público e qualquer um pode participar enviando transações, assim como, contribuindo processo de consenso (BASHIR, 2017). Elas não possuem qualquer restrição de acesso. Nela ninguém é dono do *ledger* e geralmente a rede atrai participantes oferecendo incentivos financeiros para aqueles que a mantiverem, tal como foi descrito nos algoritmos de consenso.

2.5.0.2 Privada

Este tipo de *blockchain* exige permissão para acessá-la e uma empresa detém o controle da rede, como o nome implica. Apenas pessoas convidadas pelos administradores da rede podem

se juntar a ela (BASHIR, 2017). Este tipo de *blockchain* é usado por empresas que querem aproveitar as vantagens do sistema distribuído mas não se sentem confortáveis com o baixo nível de controle das redes públicas.

2.5.0.3 Federada ou de Consórcio

Este é um tipo de *blockchain* que exige permissão para participar, porém não é controlada por uma entidade única (HÖLBL et al., 2018). Diferentes organizações compartilham a mesma rede e tem igual poder de decisão. É considerada uma rede semi-descentralizada. Além disso, elas podem ser parcialmente abertas ao público porém garantindo privilégios às organizações participantes do "consórcio".

2.6 Criptomoedas

Criptomoedas são moedas digitais descentralizadas. Elas são atualmente a principal aplicação da tecnologia *blockchain*. Segundo LANSKY, criptomoedas são "o único tipo de moeda com as três características a seguir: garantia de pseudo-anonimato, independência de uma autoridade central e proteção ao problema de duplo gasto"(LANSKY, 2018, p.3). É evidente como moedas baseadas em *blockchain* logram alcançar tais características tendo em vista as propriedades expostas na subseção 2.2 e os problemas resolvidos pelo consenso distribuído. As principais criptomoedas atualmente são o Bitcoin, o Ether (do Ethereum) e o Ripple.

2.7 Contratos Inteligentes

Idealizado por Szabo (1997), um contrato inteligente (*smart contract*) é um código determinístico executado por computadores distintos que concordam com o resultado da execução. Assim, os contratos inteligentes operam como atores autônomos, cujo comportamento é completamente previsível (Christidis; Devetsikiotis, 2016). Os contratos inteligentes são usados para permitir interações entre partes que não confiam umas nas outras para além de transações simples e sem depender de terceiros. Depois que as partes envolvidas aceitam o contrato inteligente, ele é executado pelos computadores. A partir daí, as partes envolvidas não podem alterar o contrato inteligente nem interferir na sua execução.

A tecnologia *blockchain* provê a infraestrutura sobre a qual rodam os contratos inteligentes (BASHIR, 2017). Ela é usada para armazenar e endereçar contratos inteligentes na forma de *scripts* executáveis e imutáveis. Ao invés da identificação por chave pública, um contrato inteligente geralmente é identificado por um *hash*. Uma transação ligada a um contrato inteligente aciona sua execução pelos participantes da rede *peer-to-peer*. Enquanto está sendo executado, um contrato inteligente pode mudar seu estado e originar novas transações. O histórico de estados de

um contrato inteligente, bem como as transações que ele recebeu e enviou, são permanentemente armazenados na *blockchain*.

Nem todos os *blockchains* foram projetados para suportar contratos inteligentes, como é o caso do Bitcoin. Já o Ethereum fornece um ambiente de execução Turing-completo chamado Ethereum Virtual Machine para executar contratos inteligentes (WOOD, 2014). Para facilitar a elaboração de contratos inteligentes, linguagens de alto nível que compilam essas instruções específicas de *bytecode* podem ser usadas, por exemplo, Solidity. Os contratos inteligentes baseados em Solidity contêm funções e variáveis. Por exemplo, o construtor inicializa variáveis dentro do contrato inteligente e é executado uma vez quando o contrato inteligente é implantado, isto é, criado no *blockchain*.

Para proteger o *blockchain* do Ethereum de abusos e recompensar a rede pela execução de contratos inteligentes, são aplicadas taxas. O endereço que deseja ter um contrato inteligente executado pela emissão de transação para o contrato inteligente deve pagar as taxas de execução. Essas taxas são definidas em termos de *gas*. O *gas* é a unidade de custo computacional do Ethereum (BUTERIN et al., 2014). As taxas de execução começam a partir de 2100 *gas*. Além disso, é preciso pagar por todas as instruções a serem executadas no contrato inteligente. Por exemplo, executar uma instrução de adição custa cinco *gas*. Os usuários do Ethereum podem pagar com Ether, a criptomoeda do Ethereum. Quanto mais o Ether oferecer por *gas*, mais rápido sua transação será processada pela rede.

2.8 Tokens

Tokens oferecem outras funcionalidades para além de ser um meio de troca de propósito geral tal como as criptomoedas. *Tokens* comumente são emitidos numa estrutura de *Initial Token Offering* (Oferta Inicial de Tokens) ou "*ITO*" - frequentemente referido na mídia e na literatura como "*ICO*"(*Initial Token Offering*) - para levantar fundos para um dado projeto ou empreendimento (HOUBEN; SNYERS, 2018). Eles constituem uma nova classe de ativos digitais que surgem do uso da tecnologia *blockchain* e que incorporam algum tipo de direito sobre uma entidade, ou sobre seus fluxos de caixa, ativos, bens ou serviços futuros, etc.

Alguns *tokens* assemelham-se a mecanismos tradicionais como ações ou títulos de empresas e são comumente referidos como "*security tokens*" (*tokens* de seguridade) ou *"investiment tokens"* (*tokens* de investimento) (ROHR; WRIGHT, 2018). Outros *tokens* garantem aos seus portadores acesso a produtos ou serviços específicos e são comumente referidos como *"utility tokens"* (*tokens* utilitários). Eles podem ser usados para adquirir certos produtos ou serviços, porém não constituem um meio de troca de propósito geral, simplesmente porque eles geralmente podem apenas serem usados dentro da sua própria aplicação (ROHR; WRIGHT, 2018).

2.9 Ludificação

A ludificação é definida como a utilização, em situações do mundo real, de elementos tradicionalmente encontrados nos jogos digitais, como narrativa, sistema de recompensas, competição, entre outros, com o objetivo de tentar obter um grau de envolvimento e motivação no mesmo nível que encontramos em jogadores quando interagem com bons jogos digitais (FARDO, 2013).

O estudo de Hamari et al. (2014) sugere que a ludificação de fato funciona mas com ressalvas. A maioria dos trabalhos revisados por esse estudo obtiveram resultados positivos da ludificação, mas esses efeitos foram obtidos apenas em parte dos elementos de ludificação, reforçando o papel do contexto na eficácia do método.

A título de exemplo, o artigo de Denny (2013) traz um experimento controlado em larga escala, com mais de mil participantes, medindo o impacto da incorporação de um sistema de recompensas baseado em medalhas dentro de uma ferramenta de aprendizado online. Ele informa um efeito altamente positivo tanto na quantidade de tarefas realizadas pelos alunos, sem uma respectiva redução na qualidade, quanto na quantidade de tempo no qual os alunos se engajaram na ferramenta. Por fim, os estudantes apreciaram a possibilidade de ganhar medalhas virtuais e indicaram uma forte preferência por tê-las disponíveis na interface de usuário.

3
Materiais e Métodos

Neste capítulo serão apresentadas as tecnologias a serem utilizadas no desenvolvimento da aplicação e as justificativas de suas escolhas. Posteriormente, será detalhado o método usado no processo de desenvolvimento do sistema.

3.1 Materiais

Nesta seção serão descritos os materiais utilizados na realização deste trabalho.

3.1.1 Ethereum

Ethereum é uma plataforma pública e de código aberto, lançada em 2015, que é a pioneira no suporte a contratos inteligentes, permitindo a execução de uma máquina de Turing completa (BUTERIN et al., 2014). Ela provê uma plataforma computacional com uma máquina virtual descentralizada denominada *Ethereum Virtual Machines* (EVM), que executa contratos usando uma criptomoeda denominada *ether* (BUTERIN et al., 2014). No Ethereum, os contratos inteligentes são escritos em linguagens de programação como Solidity e Serpent (derivação do Python).

Ela é atualmente a principal plataforma de projetos que utilizam contratos inteligentes e *tokens*, e possui uma comunidade ativa de desenvolvedores (FILHO; BRAGA; LEAL, 2016). Além disso, há material didático disponível de fácil acesso na internet. Por conta desses fatores e como as características dessa plataforma - tais como taxa de processamento de transações, protocolo de consenso, etc - foram consideradas adequadas ao projeto, optou-se por desenvolver o projeto utilizando essa plataforma, em detrimento de se criar uma plataforma própria para o projeto.

Além da rede pública principal do Ethereum, existem algumas redes públicas de teste para que desenvolvedores possam desenvolver, instanciar e operar seus *smart contracts* sem

gastar *ethers* reais. Elas apenas utilizam *ethers* falsos para simular de custos. Este trabalho utilizou a rede de testes *Rinkeby*, que utiliza o protocolo de consenso *Proof-of-Authority*.

3.1.2 Solidity

Solidity é uma linguagem de domínio específico para programação de contratos no Ethereum. Existem, no entanto, outras linguagens, como *Serpent*, *Mutan* e *LLL*, mas *Solidity* é a mais popular no momento. Sua sintaxe é semelhante ao *JavaScript* e ao C. *Solidity* também é chamada de linguagem orientada a contratos. No *Solidity*, os contratos são equivalentes ao conceito de classes em outras linguagens de programação orientadas a objeto (BASHIR, 2017).

É uma linguagem tipada estaticamente, o que significa que a verificação do tipo de variável em *Solidity* é realizada em tempo de compilação. Cada variável, estado ou local, deve ser especificada com um tipo em tempo de compilação. Isso é benéfico no sentido de que qualquer validação e verificação é concluída em tempo de compilação e certos tipos de erros, como interpretação de tipos de dados, podem ser detectados anteriormente no ciclo de desenvolvimento em vez de no tempo de execução, o que poderia ser caro, especialmente no caso do paradigma de *blockchain* e de contratos inteligentes (BASHIR, 2017). Outros recursos da linguagem incluem herança, bibliotecas e a capacidade de definir tipos de dados compostos.

Essas características foram consideradas adequadas à realização do projeto e, por conseguinte, essa foi a linguagem escolhida para a implementação do *token* e dos contratos inteligentes necessários. Para a programação em Solidity foi utilizada a IDE *Remix*.

3.1.3 React

Para o desenvolvimento da interface *Web* do projeto, foi escolhida a biblioteca React, que usa a linguagem *Javascript* (INC., 2019). Ele é uma das principais ferramentas de desenvolvimento *front-end* do mercado e vêm se mantendo com uma popularidade estável entre os desenvolvedores ao longo dos anos, como se pode observar no gráfico da Figura 5. Ele oferece diversas funcionalidades que facilitam a construção de websites, como responsividade, interface REST, validação de formulários, formulários reativos, entre outros. A programação da interface *Web* foi feita com o editor de texto *Visual Studio Code*. Para a comunicação da aplicação *Web* com a *blockchain* Ethereum foi utilizada a API *Web3.js*. Além disso, junto com o React foi feito uso do framework Next.js. Next.js é um *framework* que permite criar aplicações React renderizadas em servidor, o que permite mais rapidez no carregamento de páginas.

3.1.4 Metamask

O *MetaMask* é uma extensão de navegador para gerenciamento de contas Ethereum (METAMASK, 2019). Ele é compatível com os navegadores *Chrome, Firefox, Opera* e *Brave*. Ele faz a comunicação com nós Ethereum, sendo um provedor para a API Web3.js, e também

Figura 5 – Popularidade dos principais *frameworks Web* nos últimos 5 anos.

Fonte: <https://trends.google.com/trends/explore?date=today%205-y&geo=US&q=React,Angular,Vue.js>.

funciona como uma carteira digital. A aplicação Web desenvolvida acessa o MetaMask para obter a conta a ser utilizada e submeter transações a partir dela. Ele utiliza a infraestrutura Infura para permitir submeter transações sem precisar manter localmente um nó da *blockchain*.

3.1.5 Infura

Infura é uma infraestrutura de *back-end* escalável para criar aplicações na blockchain Ethereum (INFURA, 2018). É um método para conectar-se à rede Ethereum sem precisar manter localmente um nó da blockchain. O *Infura* é uma coleção de nós completos do Ethereum, tanto da rede principal quanto das redes de teste, que permitem que as aplicações se conectem a esses nós através de sua interface. Fez-se uso do plano gratuito, que possui um limite de cem mil requisições diárias aos nós da blockchain, limite este que pode ser ampliado para até cinco milhões de requisições diárias nos planos pagos. Foi utilizado para que a aplicação *Web* possa ser utilizada mesmo que o usuário não esteja mantendo um nó da *blockchain* localmente.

3.2 Métodos

O presente trabalho foi conduzido para construção de um Mínimo Produto Viável, ou MVP (sigla do inglês *Minimum Viable Product*), através do método Ludus. O MVP é uma versão do produto com apenas recursos o suficiente para que o produto tenha valor para os usuários e que permita a obtenção de reações e medições para orientar o desenvolvimento futuro (RIES,

2011). Ele possibilita uma volta completa pelo ciclo de construção, medição e aprendizagem, com o mínimo de esforço e o menor tempo de desenvolvimento.

O método Ludus (BARRETO, 2019) foi criado com o objetivo de viabilizar a construção de um protótipo de um produto utilizando-se como base o objeto de pesquisa tratado no Trabalho de Conclusão de Curso (TCC) do aluno, a justificativa para utilização deste método e desenvolvimento do MVP é explorar ao máximo o esforço despendido na realização da pesquisa e, com isso, impactar a sociedade, academia e pesquisador com um produto inovador, oriundo da pesquisa científica, e rentável. Este método trata-se de uma abordagem que mescla gestão de projetos e engenharia de software construída para atender as especificidades da realidade acadêmica brasileira, mais especificamente o grupo de pesquisa LUDIICO (Laboratório para Universalização do Desenvolvimento, Inovação e Inteligência Computacional), situado no Departamento de Computação da Universidade Federal de Sergipe. Os valores visados nó método são planejamento, objetividade e flexibilidade na condução. O método evolui em seu processo da seguinte forma:

1. Fase de definição - nesta etapa o aluno trata de definir o objetivo do MVP e como se dará sua associação com o TCC;
2. Fase de Planejamento - nesta etapa é realizado o trabalho mais pertinente a engenharia de software: a partir do objetivo criado o escopo do projeto é definido e quebrado em tarefas de desenvolvimento;
3. Fase de criação do Backlog - nesta etapa as tarefas recebem datas de entrega tendo em vista o prazo disponível para finalização do projeto;
4. Fase cíclica de gestão do projeto - nesta fase as tarefas vão sendo conduzidas semanalmente seguindo a lógica:
    a) Tarefas da semana ou *sprint backlog*;
    b) Tarefas em andamento;
    c) Validação e teste;

A fase é cíclica porque quando uma tarefa não é validada ela é alocada para uma outra semana após reavaliação. Para mais informações sobre o método Ludus, acesse o *link*[1].

Ao final do desenvolvimento do MVP, inicialmente havia sido planejado um período de testes no qual um grupo de professores e alunos iriam utilizá-lo durante o período letivo. Após esse período, seria feita uma avaliação do produto a partir da escala Likert. Como o MVP foi finalizado apenas após o término do período letivo, não foi possível realizar o período de testes. Ao invés disso, foi aplicado um questionário *online* a alunos e professores universitários

[1] <https://github.com/hugodbarreto/ludus>

para a validação do modelo de negócio a partir da escala Likert. A escala Likert é um tipo de escala de resposta psicométrica usada habitualmente em questionários para pesquisas de opinião. Ao responderem a um questionário baseado nesta escala, os perguntados especificam seu nível de concordância com uma afirmação. Esta escala tem seu nome devido à publicação de um relatório explicando seu uso por Rensis Likert (LIKERT, 1932). Na Figura 6 temos um exemplo de resposta em escala tipo Likert.

Figura 6 – Exemplo de resposta em escala tipo Likert.

Discordo Totalmente	**1**	**2**	**3**	**4**	**5**	**Concordo Totalmente**

Fonte: (DALMORO; VIEIRA, 2014).

Além disso, foi feito um levantamento acerca de alunos que poderiam ser beneficiados pelo projeto.

3.2.1 Trabalho Proposto

Propõe-se desenvolver um sistema escalável implementado na plataforma de *blockchain* Ethereum, utilizando uma rede de *blockchain* privada, com o objetivo de permitir o acúmulo de pontos sobressalentes nas disciplinas da universidade na forma de *tokens*, os "Ludicoins", para serem usados por alunos e professores do ambiente acadêmico. Dentro deste sistema o aluno poderá guardar pontos extras que sobrarem em unidades das disciplinas e utilizá-los em unidades posteriores ou em outras disciplinas de professores participantes da rede. Dessa forma, alunos e professores poderão manter um histórico de desempenho acadêmico a nível de atividades realizadas.

O sistema operará sobre uma *blockchain* privada. Sendo assim, alunos e professores que tiverem acesso permitido poderão participar como nós na rede. Um aluno será premiado com Ludicoins quando a soma das notas recebidas por prova e atividades, submetidas no sistema por um professor, de uma mesma disciplina, numa mesma turma e numa mesma unidade excederem a nota máxima 10. Depois disso, os alunos poderão utilizar os *tokens* acumulados em transações por notas em outras matérias. Cada disciplina terá uma equivalência diferente entre pontos e Ludicoins. Essas regras de transações de *tokens* serão implementadas por meio de *smart contracts* que serão executados na *blockchain*. Os usuários realizarão ações com os *smart contracts* a partir de uma aplicação *Web* integrada.

Em complemento ao desenvolvimento do sistema, deverá ser feita a validação da proposta e das regras de negócio do projeto pelo público-alvo a partir da aplicação de questionário.

3.2.2 Modelagem do Software

Nesta seção serão apresentadas as descrições e representações de diversos aspectos do projeto de software. Como o método Ludus é iterativo e incremental, a modelagem do projeto foi confeccionada ao longo do processo de desenvolvimento. Primeiramente, serão apresentadas as definições; em seguida será apresentado o documento de requisitos e, por fim, a modelagem e arquitetura da aplicação.

3.2.2.1 Definições, acrônimos e abreviaturas

Abaixo estão listados os termos e suas definições utilizados na descrição do sistema:

- **Professor**: Pessoa que leciona uma ou mais Disciplinas em uma ou mais Turmas;
- **Aluno**: Pessoa que atende a uma ou mais Turmas;
- **Disciplina**: Componente curricular acadêmica que os professores lecionam;
- **Turma**: Conjunto de alunos vinculados a um Professor e a uma Disciplina;
- **Unidade**: Conjunto de Atividades vinculado a uma Turma;
- **Atividade**: Evento ou ação que o Professor pode atribuir a uma Unidade;
- **Atividade Realizada**: Avaliação do Professor referente a uma Atividade realizada por um Aluno;
- **Ludicoin**: *Token* de *blockchain*.

3.2.2.2 Especificação de Requisitos do Sistema

Nesta subseção serão descritos os requisitos funcionais e não funcionais do sistema.

a) **Requisitos Funcionais**

RF1. Conexão com o MetaMask: A plataforma Web deve conectar-se com a extensão *MetaMask* ativa no navegador para obter o endereço da conta *Ethereum* do usuário e poder assinar transações a partir dela.
RF2. Tipos de Acesso: As funcionalidades da plataforma web devem depender do tipo de acesso do usuário. O sistema possui dois tipos de acesso: professor e aluno.
RF3. Login na Plataforma Web: Os usuários devem ter acesso a interface web via *MetaMask*. Caso o login não seja permitido, o sistema deve informar ao usuário que ele não está cadastrado ou que o *MetaMask* não está ativado.
RF4. Cadastro de Aluno: O Sistema deve permitir que o aluno possa se cadastrar no Sistema.

RF5. Solicitação de Cadastro de Professor: O Sistema deve permitir que o Professor solicite cadastro no Sistema.

RF6. Aprovação de Cadastro de Professor: O Sistema deve permitir que Professores já cadastrados possam aprovar a solicitação de cadastro de um novo professor.

RF7. Professor inicial: O Sistema deve permitir que o primeiro Professor possa se cadastrar sem precisar ser aprovado.

RF8. Adicionar de Disciplina: O Sistema deve permitir que o Professor adicione uma Disciplina.

RF9. Adicionar de Turma: O Sistema deve permitir que o Professor adicione uma Turma vinculada a uma Disciplina.

RF10. Adicionar de Atividade: O Sistema deve permitir que o Professor adicione uma Atividade vinculada a uma Turma.

RF11. Solicitação de matrícula na turma: O Sistema deve permitir que o Aluno solicite matrícula na Turma.

RF12. Aprovação de matrícula na turma: O Sistema deve permitir que o Professor aprove a matrícula de Aluno na Turma.

RF13. Armazenamento de Informações: Todas as informações cadastradas devem ser armazenadas em uma *blockchain* Ethereum privada.

RF14. Alteração de Disciplina: O Sistema deve permitir que o Professor altere o cadastro de uma Disciplina.

RF15. Alteração de Turma: O Sistema deve permitir que o Professor altere o cadastro de uma Turma.

RF16. Remoção de Disciplina: O Sistema deve permitir que o Professor remova uma Disciplina que não tenha turmas vinculadas a ela.

RF17. Visualização de Disciplinas: O Sistema deve listar todas as Disciplinas cadastradas.

RF18. Visualização de Turmas: O Sistema deve listar todas as Turmas cadastradas de forma automática.

RF19. Visualização de Atividades: O Sistema deve listar todas as Atividades cadastradas vinculadas a uma certa Turma de forma automática.

RF20. Visualização de Turmas do Professor: O Sistema deve listar todas as Turmas vinculadas a um Professor de forma automática.

RF21. Visualização de Turmas do Aluno: O Sistema deve listar todas as Turmas nas quais um Aluno está matriculado de forma automática.

RF22. Atribuição de Notas: O Professor pode atribuir uma nota a um Aluno por uma Atividade.

RF23. Visualização de Notas dos Alunos em uma Atividade: O Sistema deve listar as notas de todos os Alunos matriculados em uma determinada Atividade de forma automática.

RF24. Visualização de Notas dos Alunos em uma Turma: O Sistema deve listar as notas de todas as Unidades de todos os Alunos matriculados em uma Turma de forma automática.

RF25. Visualização de Notas do Aluno nas Atividades: O Sistema deve listar as notas do Aluno nas Atividades de determinada Turma de forma automática.

RF26. Visualização de Notas do Aluno nas Turmas: O Sistema deve listar as notas do Aluno em todas as Unidades das Turmas nas quais eles está matriculado de forma automática.

RF27. Atribuição de Ludicoins: O Sistema deve atribuir Ludicoins de forma automática a um Aluno quando, em uma Turma em que ele está matriculado, o somatório das notas dele nas Atividades de uma Unidade for maior que 10.
RF28. Gasto de Ludicoins: O Sistema deve permitir que o Aluno utilize seus Ludicoins acumulados para aumentar a nota de uma Unidade.
RF29. Cotação de Ludicoins em uma Disciplina: O Sistema deve permitir que o Professor defina qual a taxa de câmbio entre pontos de uma Disciplina e Ludicoins.

b) **Requisitos Não-Funcionais**

RNF1. A aplicação web deverá possuir um design responsivo.
RNF2. A aplicação web deverá ser compatível com os navegadores compatíveis com o *MetaMask* (são eles: *Google Chrome, Firefox, Opera* e *Brave*).
RNF3. A aplicação web deverá mostrar mensagens de erro sempre que houver falha na execução das tarefas do usuário.
RNF4. O Sistema deve permitir que o Aluno gaste Ludicoins equivalentes a até no máximo 3 pontos em uma mesma Unidade.

3.2.2.3 Casos de Uso

Nesta subseção são apresentados os atores e os casos de uso do sistema. O Diagrama de Casos de Uso é apresentado na Figura 7. O sistema apresenta 2 tipos de atores: Aluno e Professor, definidos na subseção 3.2.2.1.

3.2.2.4 Arquitetura do Projeto

Nesta subseção são apresentados a estrutura e os componentes necessários para a construção do sistema, assim como as inter-relações que ocorrem entre esses componentes. A Figura 8 ilustra a arquitetura do sistema, nota-se que ele é dividido em 4 componentes principais. A seguir serão detalhadas as estruturas de cada um desses componentes.

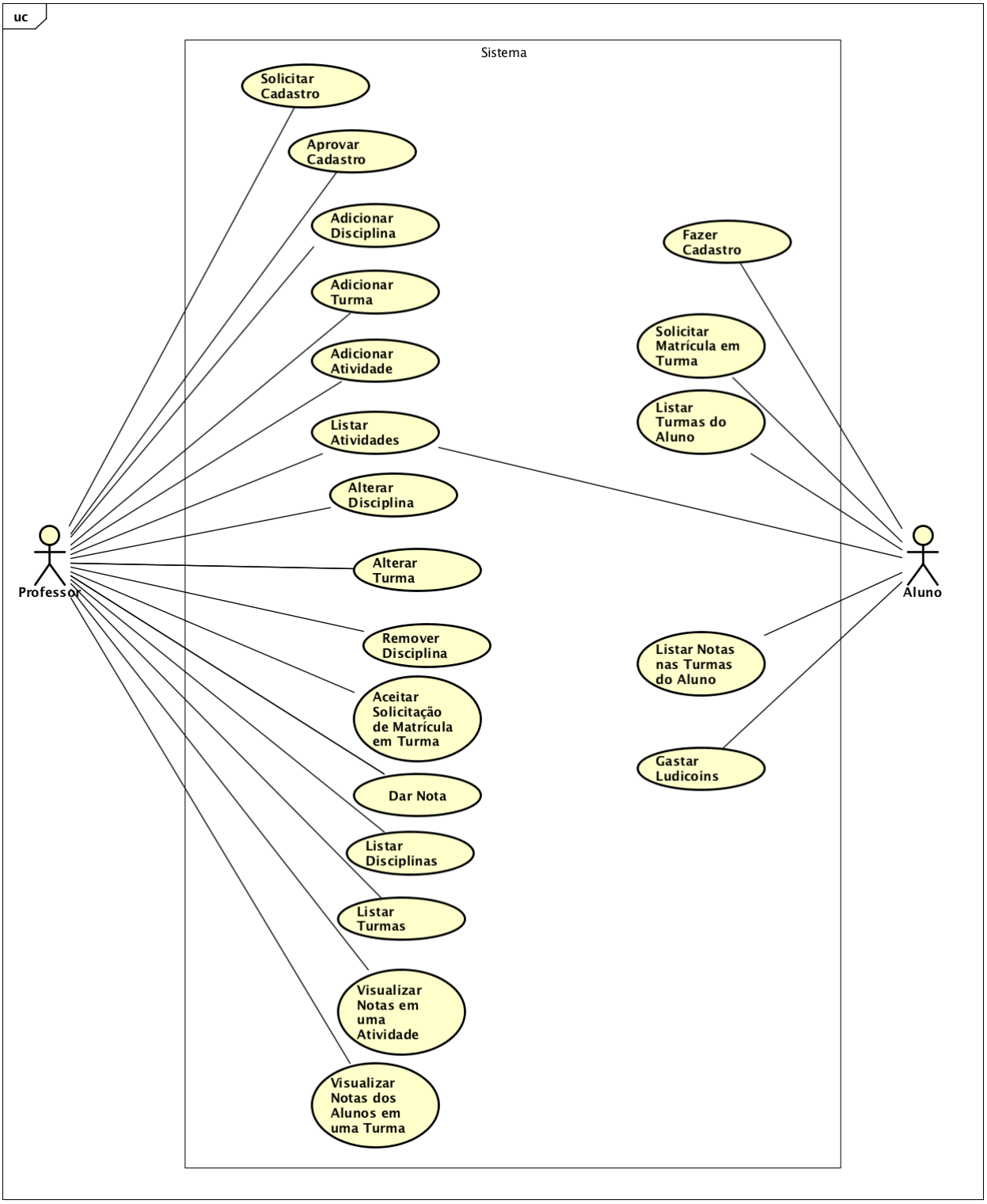
a) **Servidor**

O servidor nessa arquitetura serve apenas para armazenar a aplicação web e prover o conteúdo estático do *website*, ou seja, os documentos HTML e os *scripts* Javascript gerados pelo React. Pelo *framework* Next.js, as páginas são renderizadas já no servidor. Neste projeto foi utilizado o *localhost* como servidor dos componentes *Web*.

b) **Aplicação Web**

A aplicação web foi feita utilizando o React, que é uma biblioteca JavaScript para construção de interfaces de usuário. Cada página Web da aplicação foi criada a partir de uma

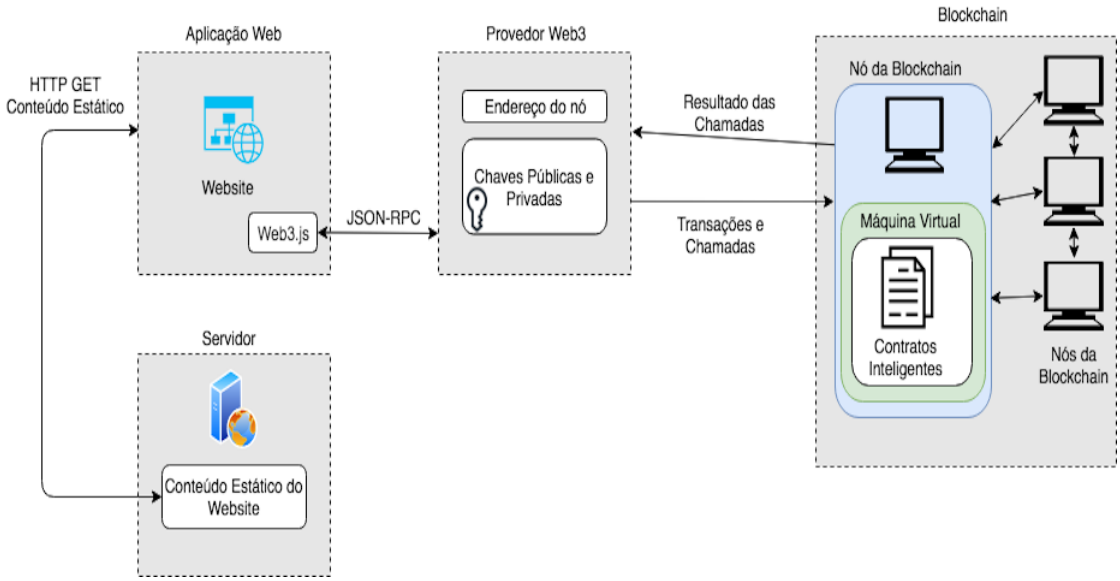
Figura 7 – Diagrama de Casos de Uso.



Fonte: Próprio autor.

classe em JavaScript que inclui elementos de exibição, como código HTML, CSS e descrições de interface em JavaScript, com a função de definir o conjunto de elementos de tela que o React irá manipular de acordo com lógica e os dados do programa. Dentro dessas classes JavaScript de

Figura 8 – Arquitetura do Sistema.



Fonte: Próprio autor.

criação de páginas, há também as funções que definem o comportamento das páginas.

Além disso, ao longo das páginas web foram usados componentes dos React. Componentes permitem você dividir a UI em partes independentes e reutilizáveis, e pensar em cada parte isoladamente. Eles podem ser tanto funções quanto classes, em JavaScript.

A aplicação Web se comunica com os *smart contracts* a partir da API Web3.js. No processo de comunicação, a API Web3.js envia requisições JSON-RPC (JSON *Remote Procedure Call*) para um provedor Web3 e a partir deste são enviadas as transações ou requisições para o nó da *blockchain* Ethereum (ETHEREUM, 2019b).

Os contratos inteligentes são instanciados na aplicação web a partir de uma função da API Web3.js que gera um objeto a partir de um arquivo JSON e do endereço do contrato. Este arquivo JSON utilizado contém o *bytecode* do código Solidity compilado e a especificação ABI (*Application Binary Interface*) do contrato. A ABI do contrato contém a interface do contrato, que descreve as assinaturas dos métodos, e é a maneira padrão de interagir com contratos no ecossistema Ethereum, tanto de fora da *blockchain* quanto em interações contrato a contrato.

c) **Provedor Web3**

O provedor Web3 faz a comunicação da API Web3 com a *blockchain* Ethereum. Ele recebe requisições JSON-RPC da API e retorna a resposta. Ele se comunica com o Ethereum através de requisições HTTP, quando o nó da *blockchain* é remoto, ou IPC, quando o nó se encontra no sistema de arquivos local. Ele também pode conter as informações de chaves públicas e privadas que identificam uma conta na *blockchain*, para poder submeter transações.

Foi utilizado o MetaMask como principal provedor Web3 no sistema. Por meio dele é obtida a conta do usuário, a partir da qual serão feitas as transações.

Para a requisição de informações sem restrição de acesso na *blockchain*, onde não é necessário saber dados da conta do usuário, é utilizado o provedor Web3 do Infura para agilizar a renderização das páginas no servidor.

d) **Blockchain**

O principal componente dessa aplicação é a *blockchain*, pois nela são armazenadas e processadas todas as informações para o funcionamento do sistema. Isso é feito a partir dos contratos inteligentes implantados nela. Eles contêm a lógica de negócio e as estruturas de dados utilizadas para armazenar as informações sobre as quais os sistema vai operar.

A rede de *blockchain empregada* foi a rede pública de teste Ethereum chamada Rinkeby. A rede Rinkeby faz uso do protocolo de consenso *Proof-of-Authority*, com uma taxa de criação de blocos de 15s.

Os *smart contracts* são armazenados no Ethereum a partir de um endereço *hash* gerado deterministicamente com a função de *hash* Keccak-25 (CARVER; MERRIAM, 2018), tendo como entradas principais o endereço, que é o número da conta, de quem submete o contrato e o número de transações submetidas por essa conta. Nesse endereço gerado é armazenado o *bytecode* do contrato. Os *smart contracts* são executados sobre a máquina virtual do Ethereum, a EVM (*Ethereum Virtual Machine*), que é Turing completa e contém um conjunto de instruções próprio (ETHEREUM, 2019a).

As interações com os contratos inteligentes podem se dar a partir de chamadas (do inglês, *calls*) ou transações. Chamadas invocam funções que não modificam dados e apenas retornam dados. Elas são executadas automaticamente e não precisam ser transmitidas para todos os nós da *blockchain*. Já transações chamam funções no contrato que modificam dados e tem um custo em *gas* associado, que é cobrado à conta que submeteu a transação. São transmitidas para todos os nós da *blockchain* e são executadas apenas quando um bloco que contém elas é minerado.

O nó da rede onde chamadas e funções são executadas é selecionado pelo provedor Web3 utilizado, podendo ser MetaMask ou Infura. O Infura contém diversos nós nas redes de *blockchain* Ethereum públicas, neste caso a utilizada é a rede Rinkeby, e provê uma camada de abstração para endereço do nó. Dessa forma, a aplicação contém um endereço do projeto no Infura e, então, o Infura redireciona para algum nó do Ethereum de forma a fazer o balanço de carga.

Por padrão, o MetaMask também faz uso do Infura para acessar o Ethereum. Ele pode ser configurado para a utilização de redes privadas, podendo fazer uso de um nó local ou remoto definido. Como neste projeto foi utilizada a rede pública Rinkeby, foi mantida a configuração padrão de usar o Infura para se conectar à rede.

3.2.2.5 Visão Lógica

Com base no modelo de visão '4+1' de Kruchten (1995), a seguir serão apresentados diagramas e descrições referentes à visão lógica do sistema.

O funcionamento do sistema pode ser visualizado em dois níveis: a nível de contratos inteligentes e a nível de entidades do domínio. A visão a nível de contratos inteligentes informa as relações entre os contratos. A visão a nível de entidades do domínio da aplicação informa como as entidades, identificadas a partir das regras de negócio, se relacionam entre si.

A implementação de *smart contracts* em Solidity segue um paradigma chamado de programação orientada a contratos. Nesse contexto, contratos são equivalentes ao conceito de classes no paradigma de Orientação a Objetos. Sendo assim, pode-se construir um diagrama de classes a partir dos contratos do projeto. A Figura 9 apresenta as relações entre os contratos implementados.

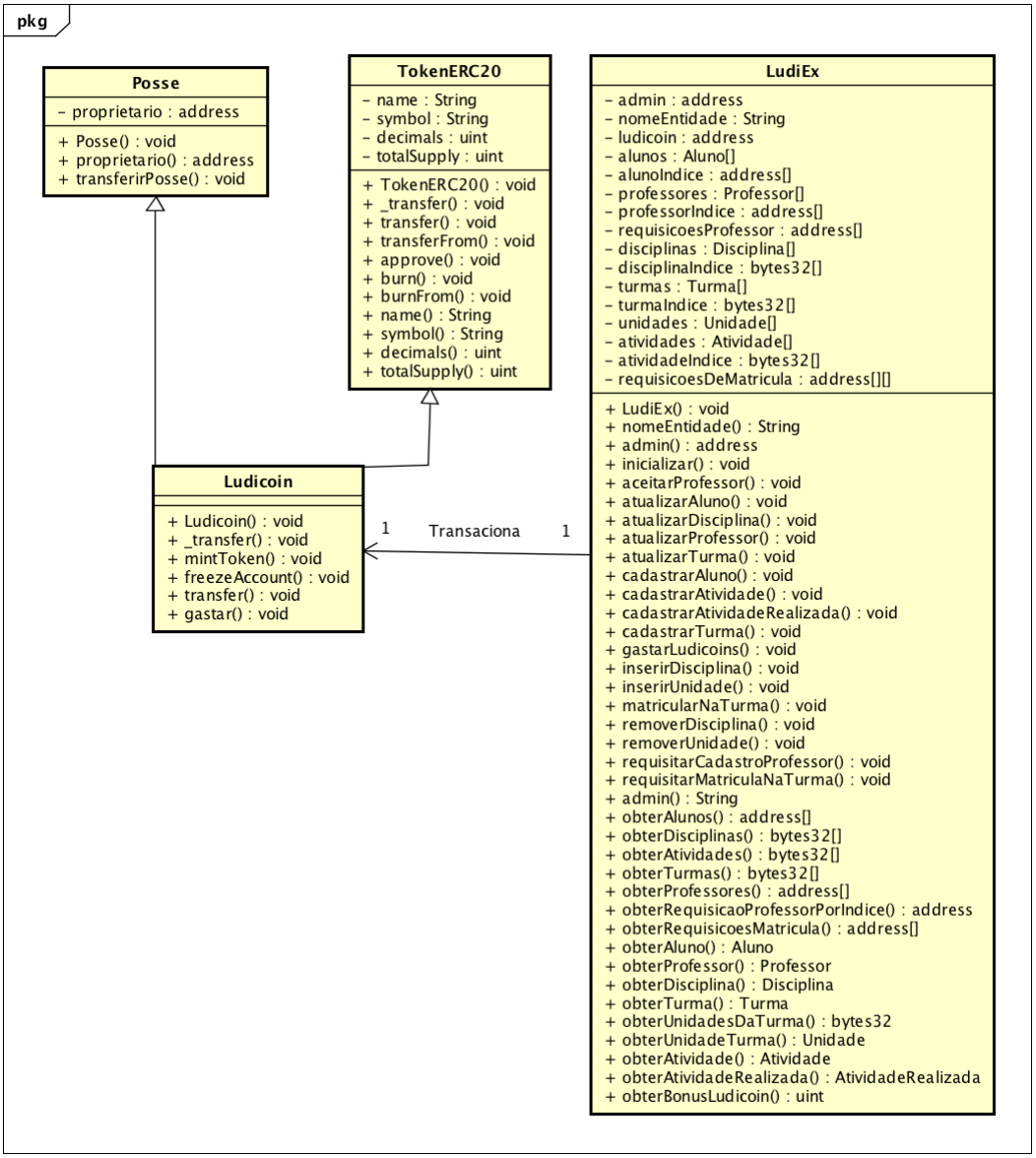
O contrato Posse guarda o atual "proprietário"do contrato e define uma função de restrição que permite acesso somente ao proprietário. O contrato TokenERC20 implementa funções no padrão de *token* Ethereum ERC-20 (atualmente chamado de EIP-20). O ERC-20 é uma interface padrão para *tokens* Ethereum, o que permite compatibilidade com as principais carteiras Ethereum (VOGELSTELLER; BUTERIN, 2019). Este padrão descreve o funcionamento básico para transferência de *tokens* e verificação de saldo, além de quais eventos devem ser disparados na *blockchain*. O Código 1 mostra a interface ERC-20. Para facilitar o uso em carteiras Ethereum também é recomendada a implementação de funções de acesso para os atributos *name*, *decimals* e *symbol*, que informam o nome, a quantidade de casas decimais e o símbolo, respectivamente.

Código 1 – Interface do padrão ERC-20

```
1  contract ERC20Interface {
2      function totalSupply() public view returns (uint);
3      function balanceOf(address tokenOwner) public view returns (uint balance);
4      function allowance(address tokenOwner, address spender) public view returns (uint
       ↪  remaining);
5      function transfer(address to, uint tokens) public returns (bool success);
6      function approve(address spender, uint tokens) public returns (bool success);
7      function transferFrom(address from, address to, uint tokens) public returns (bool
       ↪  success);
8
9      event Transfer(address indexed from, address indexed to, uint tokens);
10     event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
11 }
```

O contrato Ludicoin define o *token* Ludicoin, que é o *token* Ethereum a ser transacionado pelo sistema. Ele herda os contratos Posse e TokenERC20. Além disso, ele implementa as

Figura 9 – Diagrama de Classes dos Contratos.



Fonte: Próprio autor.

funções: *mintToken*, para a emissão de novos *tokens*; a função *freezeAccount*, para congelar uma conta; *transfer*, que sobrescreve a função da classe TokenERC20 para permitir que somente o proprietário do contrato possa trasferir *tokens*; e gastar, que permite que o usuário gaste Ludicoins nas notas das disciplinas. Na implantação do contrato a "posse" deste é transferida para o contrato LudiEx, fazendo com que só ele tenha permissão para gerar e transacionar Ludicoins.

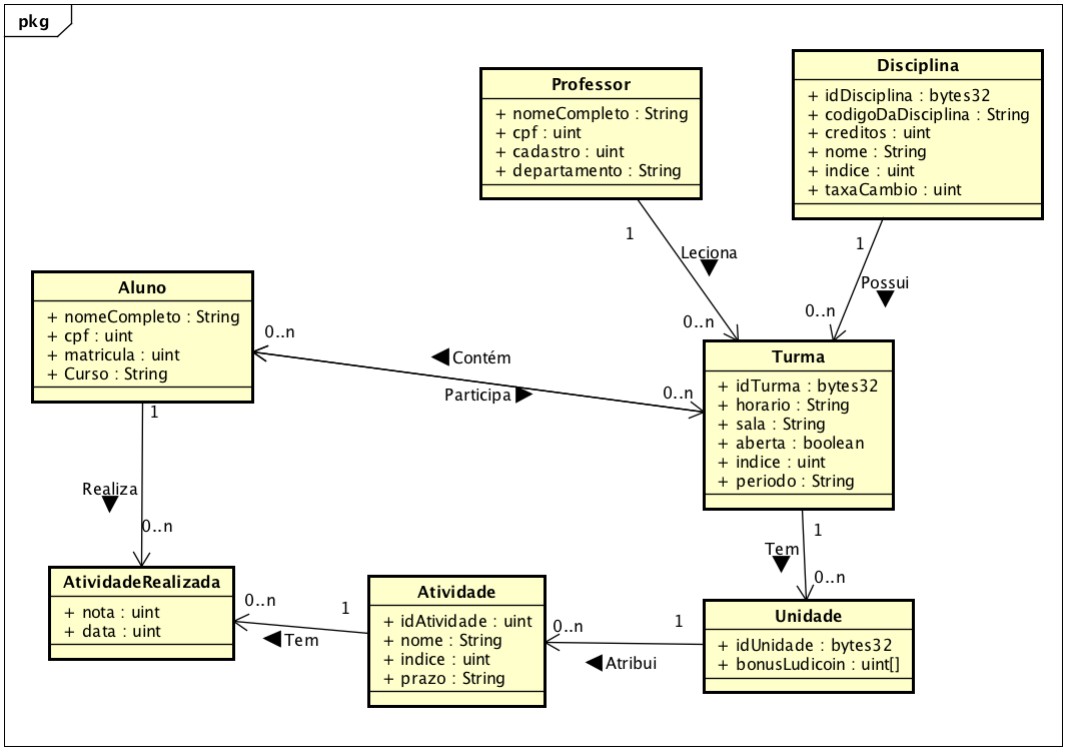
O contrato LudiEx contém todas as regras de negócio do sistema. Ele implementa as operações de CRUD (*Create-Remove-Update-Delete*) das entidades do sistema e estabelece as relações que que permitem suas duas funcionalidades principais: gerar Ludicoins quando um aluno obtiver uma nota acumulada maior do que a nota máxima 10 em uma unidade de uma disciplina e permitir que o aluno gaste esses *tokens* para aumentar a nota, seja em outra unidade

da mesma turma ou de outra turma.

Esses Ludicoins são gerados e gastos com base numa taxa de câmbio entre Ludicoins e pontos na disciplina. Por exemplo, 1 ponto extra numa disciplina pode valer 1,5 Ludicoins, podendo valer até no máximo 3 Ludicoins. Isso é definido pelo campo *taxaCambio* da estrutura Disciplina.

As entidades de domínio da aplicação foram implementadas como estruturas (*structs*), ou seja, tipos de dados compostos, dentro do contrato LudiEx. Mais adiante, no subtópico Desenvolvimento, será explanado o porquê dessa decisão. As 7 entidades identificadas foram: Professor, Aluno, Disciplina, Turma, Unidade, Atividade e AtividadeRealizada. As estruturas que representam as entidades do domínio da aplicação são apresentadas na Figura 10.

Figura 10 – Diagrama das Estruturas das Entidades do Domínio.



Fonte: Próprio autor.

3.2.3 Desenvolvimento

O desenvolvimento do sistema passou por duas etapas principais: confecção dos contratos inteligentes e confecção da aplicação web. Ao final do desenvolvimento, foi aplicado um questionário a alunos e professores de universidades para avaliar a proposta e o modelo de negócio; e foi feito um levantamento acerca de alunos que poderiam ser beneficiados pelo projeto. A seguir tem-se os detalhes de cada uma dessas etapas.

3.2.3.1 Contratos Inteligentes

Inicialmente, foram implementados os contratos inteligentes na IDE Remix. O Remix faz a compilação dos contratos e permite fazer a implantação deles tanto numa máquina virtual da própria da IDE como também em redes públicas Ethereum. Nessa fase inicial os contratos foram testados na máquina virtual da IDE.

Os contratos TokenERC20 e Ludicoin foram implementados em conformidade com a interface padrão ERC-20, com nomes de funções e parâmetros correspondendo aos exigidos pela interface, e baseados nos conceitos de operações básicas de *tokens* apresentadas pela documentação do Ethereum.

O contrato Posse define um "proprietário" para o contrato, uma função de transferência de "proprietário" e um *modifier* contendo um *require*. Em Solidity, *modifiers* são trechos de código que são executados antes das funções que os utilizam. São declarados nas funções junto com os modificadores de acesso (*public, private*, etc). Esse *modifier* contém um require, que é uma condição no Solidity que deve ser atendida para que a função possa ser executada. O Código 2 apresenta o código completo do contrato Posse e contém um exemplo de *modifer* junto com um exemplo de seu uso. O objetivo desse contrato é criar uma restrição de acesso reutilizável por meio de herança.

O contrato LudiEx contém toda a lógica das regras de negócio do sistema. Apesar de uma certa equivalência entre os paradigmas de Programação Orientada a Objetos e da Programação Orientada a Contratos do Solidity, os *smart contracts* ainda devem ser vistos como *scripts* cuja implementação deve ser simples.

Código 2 – Código do Contrato Posse

```
1  pragma solidity ^0.4.17;
2
3  contract Posse {
4      address public proprietario;
5      function Posse() public {
6          proprietario = msg.sender;
7      }
8      modifier somenteProprietario {
9          require(msg.sender == proprietario);
10         _;
11     }
12     function transferirPosse(address novoProprietario) somenteProprietario public {
13         proprietario = novoProprietario;
14     }
15 }
```

Numa primeira versão desse contrato, cada entidade do domínio da aplicação havia sido representada por um contrato separado, o que equivaleria a uma classe. No entanto, a declaração de instâncias desses contratos no contrato principal LudiEx fez com que o contrato LudiEx ultrapassasse 24576 bytes - o tamanho máximo de contrato permitido pelo Ethereum. Além disso, a implantação de um contrato na *blockchain* para cada objeto instanciado e a troca de mensagens entre contratos acarretariam em mais operações por transação e, portanto, custos de *gas* mais elevados por transação.

Para simplificar o projeto dos contratos inteligentes do sistema e diminuir o tamanho do contrato LudiEx, as entidades foram implementadas como estruturas (*structs*) dentro deste contrato. Dessa forma, o acesso e modificação dos dados dessas entidades foram simplificados. Mesmo assim, o tamanho final do contrato do LudiEx ficou próximo do tamanho máximo, não permitindo incrementos sem que haja uma refatoração.

Por fim, há de se destacar a ausência de banco de dados do projeto. Os dados são armazenados nos *smart contracts* e persistem enquanto a *blockchain* que os contém for mantida. Para cada estrutura no LudiEx existe um arranjo (*array*) listando as chaves de cada item e um *mapping* do Solidity, que funciona como uma tabela *hash* para o acesso dos dados de cada item. A integridade referencial entre as entidades é feita por campos nas estruturas declaradas.

3.2.3.2 Aplicação Web

A aplicação web foi desenvolvida em JavaScript utilizando a biblioteca React. Também foram aplicadas a APIs Web3.js e o *framework* Next.js. A lista completa das dependências do projeto pode ser vista na Figura 11.

Figura 11 – Dependências do projeto.

```
"dependencies": {
    "fs-extra": "^8.1.0",
    "ganache-cli": "^6.5.0",
    "mocha": "^6.1.4",
    "next": "^4.1.4",
    "next-routes": "^1.4.2",
    "react": "^16.8.6",
    "react-dom": "^16.8.6",
    "semantic-ui-css": "^2.4.1",
    "semantic-ui-react": "^0.87.3",
    "solc": "^0.4.17",
    "truffle-hdwallet-provider": "0.0.3",
    "web3": "^1.0.0-beta.55"
  }
```

Fonte: Próprio autor.

Cada página web foi criada a partir de uma classe em JavaScript que herda a classe *Component* do React. Além disso, foram criados 4 componentes React reutilizáveis aplicados ao longo das páginas: *Header, Layout, LayoutProfessor* e *MenuVertical*. As páginas foram desenvolvidas de forma que permitam que todos os casos de uso possam ser realizados.

O roteamento entre páginas foi feito a partir da API Routes do Next.js. Ela permite a passagem de valores pela URL da página. O Código 3 contém um exemplo de uso de Routes. Dois-pontos na URL indica que o nome a seguir é uma variável. O Código 4 mostra como o React recebe esses valores a partir do objeto *props*.

Código 3 – Passagem de valor a partir de URL

```
1 const routes = require('next-routes')();
2
3 routes.add('/professor/:endereco', '/professor/indexProfessor');
4
5 module.exports = routes;
```

Código 4 – Recebimento de valor a partir de props

```
1 class IndexProfessor extends Component {
2
3     static async getInitialProps(props) {
4         const conta = props.query.endereco;
```

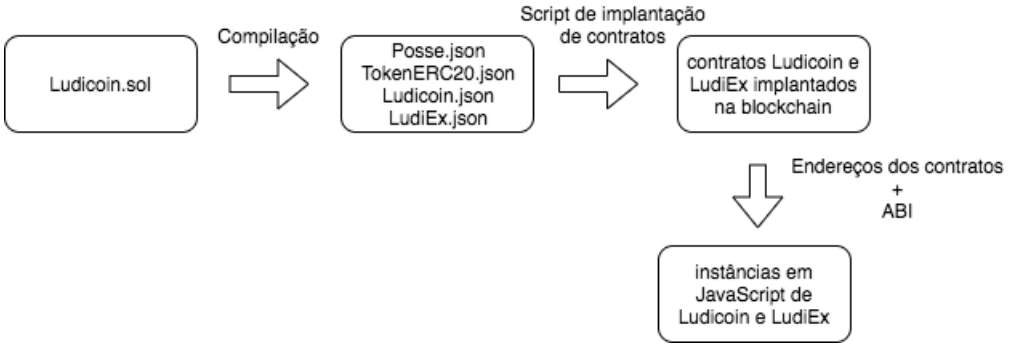
Para interagir com os contratos inteligentes a partir da aplicação web, deu-se o seguinte processo, ilustrado na Figura 12. Primeiramente, o código de todos contratos foi posto no mesmo arquivo *Ludicoin.sol* para facilitar o processo de compilação. Depois foi confeccionado um *script* no arquivo *compile.js*, que faz uso do compilador Solidity instalado no projeto para gerar arquivos JSON, um para cada contrato, contendo os respectivos *bytecodes* e ABIs (interfaces) dos contratos. Em seguida, é feita a implantação dos contratos Ludicoin e LudiEx a partir de um *script* usando a API Web3.js. Depois de feita a implantação dos contratos, é feita uma instância para cada um deles. Essa instância é criada a partir do objeto *web3.eth.Contract* que recebe o campo *interface* do arquivo JSON, que contém a ABI, e o endereço do contrato. Finalizado esse processo, essas instâncias podem ser importadas nas páginas web e utilizar os métodos de seus contratos.

Durante todo o processo de desenvolvimento da aplicação web, foi utilizada a rede pública de teste Rinkeby e o servidor da aplicação foi o próprio *localhost*.

3.2.4 Questionário e Levantamento

Inicialmente, havia sido planejada a aplicação do sistema para alunos e professores de turmas da Universidade Federal de Sergipe e, em seguida, seria aplicado um questionário para a validação do produto. Como o sistema só foi finalizado após o fim do término do período letivo,

Figura 12 – Criação de objetos em JavaScript dos contratos inteligentes.



Fonte: Próprio autor.

essas etapas não puderam ser realizadas. No lugar dessas etapas, foi aplicado um questionário *online* para alunos e professores universitários para validar a proposta e as regras de negócio. Neste questionário, o participante se identifica como aluno ou professor e, em seguida, responde a 3 itens. As respostas são dadas de acordo com o grau de concordância do participante em escala Likert, variando de 1 ("Discordo completamente") até 5 ("Concordo completamente"). Os 3 itens são:

Q1) Guardar pontos extras para usar em outras unidades ou disciplinas pode ser útil para motivar os alunos a realizarem todas as suas atividades.

Q2) "Ludicoins" devem ser gerados somente quando o aluno atingir pontuação maior que 10 em uma unidade de uma disciplina.

Q3) Pontos de diferentes disciplinas devem ter diferentes valores quando convertidos para "Ludicoins".

O primeiro item possui a intenção de validar a proposta. O segundo e o terceiro avaliam as regras de negócio atuais. Além disso, esse questionário possui um campo para comentários, sugestões e informações adicionais.

Por fim, foi feito um levantamento junto a professores do Departamento de Computação da Universidade Federal de Sergipe para buscar alunos que poderiam ter sido impactados pela implantação do Ludicoin no ano-período 2019.1. A condição de busca foi por alunos que tiraram nota 10 em turmas em que havia possibilidade pontos extras por meio de realização de atividades acadêmicas.

4

Trabalhos Relacionados

Este capítulo apresenta os métodos e resultados da busca por trabalhos relacionados. Para tal, foi utilizada a metodologia de mapeamento sistemático com foco em aplicações de *blockchain* na educação, que é a área de atuação do presente trabalho.

4.1 Objetivo do Mapeamento Sistemático

Um mapeamento sistemático (MS) é um tipo estudo secundário que faz uma revisão abrangente dos estudos primários existentes em relação a um tópico amplo de pesquisa com o objetivo de identificar e classificar a pesquisa relacionada a esse tópico. Um MS pode ajudar a identificar lacunas na área e auxiliar no posicionamento de novas atividades de pesquisa acerca do tema (KITCHENHAM; CHARTERS, 2007).

Tendo em vista esses objetivos, o presente trabalho baseia-se no método descrito por Petersen et al. (2008) para fazer um mapeamento sistemático acerca das aplicações de *blockchain* para educação. Este método é apresentado em 4 etapas principais:

- Definição das questões de pesquisa;
- Busca de estudos primários
- Seleção dos estudos relevantes;
- Extração de dados e análise de resultados.

4.2 Questões de pesquisa

Esse mapeamento sistemático tem como objetivo analisar os trabalhos publicados em periódicos revisados por pares, com o propósito de fazer o levantamento das aplicações de

blockchain na educação. Tal objetivo está norteado pelas seguintes questões:

- **Q1)** Quais são as aplicações mais recentes para promover transformações na educação por meio de *blockchain*?
- **Q2)** Como *blockchain* é usado para melhorar a educação?

4.3 Estratégias de Busca e Seleção

Os estudos primários foram destacados para seleção através do uso de *strings* de busca compostas por palavras-chave, relacionadas por lógica booleana e selecionadas para promover a procura de pesquisas que auxiliem na resposta às questões de pesquisa. Essas *strings* foram aplicadas nas plataformas de busca das bases de dados selecionadas de artigos e patentes.

Neste mapeamento foi utilizada a base de dados *SciVerse Scopus* para mapear os artigos do estado da arte. *SciVerse Scopus* é um indexador de resumos e citações de artigos para jornais e revistas acadêmicos das principais editoras, incluindo editoras como *IEEE Xplorer*, *ACM Digital Library* e *Springer Link*.

Para mapear as patentes do estado da técnica foram utilizadas as seguintes bases de dados: *Espacenet* - base de dados do *European Patent Office* (EPO), com documentos de mais de 90 países - e o Instituto Nacional de Propriedade Intelectual (INPI), que abrange os registros de patentes no Brasil.

Tendo em vista os objetivos de responder às questões de pesquisa e ser abrangente para recuperar mais estudos, os termos busca foram:

- Em inglês: *(“blockchain” OR "block chain"OR “block-chain” OR “cryptocurrency” OR “distributed ledger”) AND “education”*
- Em português: *(“blockchain” OR "block chain"OR “block-chain” OR “criptomoeda” OR “registro distribuído”) AND “educação”*

As buscas foram realizadas em 13 de março de 2019 nas bases de dados. Foi utilizado o portal de periódicos da CAPES para evitar as possíveis restrições de download das editoras indexadas pela base *Scopus*. Pela aplicação da *string* de busca na *Scopus* foram encontrados 108 resultados de estudos primários. Pode-se observar a distribuição de artigos por ano na Tabela 1.

Em seguida foram encontradas 12 patentes resultantes da busca na *Espacenet* e nenhum resultado foi encontrado pelo INPI, conforme apresentado na Tabela 2.

Depois disso, os resultados destas buscas foram filtrados através dos critérios de seleção definidos abaixo.

Tabela 1 – Artigos primários retornados pela base *Scopus* por ano.

Ano	Estudos primários retornados
2015	2
2016	4
2017	16
2018	70
2019	16
Total	**108**

Fonte: Próprio autor.

Tabela 2 – Patentes depositadas por base de dados.

Bases	Patentes depositadas
Espacenet	12
INPI	0
Total	**12**

Fonte: Próprio autor.

4.4 Critérios de Seleção

Para que o artigo ou patente seja incluído na lista de trabalhos ele deve cumprir todos os critérios de inclusão adotados referentes ao seu tipo de trabalho. Os critérios de inclusão adotados são:

- Artigos ou patentes depositadas relacionados com a aplicação de *blockchain* na educação;
- Artigos publicados em periódicos revisados por pares;
- Artigos ou patentes depositadas publicados em português ou inglês;

Já para a exclusão do artigo da seleção basta atender um dos critérios de exclusão listados a seguir:

- Artigos ou patentes depositadas inacessíveis;
- Artigos ou patentes duplicadas;
- Artigos de estudos secundários ou terciários;
- Artigos sobre o ensino de *blockchain*;

Tabela 3 – Estudos selecionados após aplicação do critério de seleção por base.

Bases	Estudos relevantes
IEEE Xplorer	9
Springer	1
ACM Digital Library	1
IAEME	1
SciTePress	1
Sciendo	1
Ledger	1
International Journal of Network Management	1
Total	**16**

Fonte: Próprio autor.

Após a aplicação dos critérios de inclusão e exclusão, foram obtidos 16 artigos relevantes dos 108 selecionados, como pode ser visto na tabela 3.

Já entre as patentes, obtiveram-se 7 patentes relevantes das 12 selecionadas, como mostrado na Tabela 4.

Tabela 4 – Patentes selecionadas após aplicação do critério de seleção.

Bases	Patentes depositadas	Patentes selecionadas
Espacenet	12	7
INPI	0	0
Total	**12**	**7**

Fonte: Próprio autor.

4.5 Resultados

Considerando o conjunto de 7 patentes e 16 artigos relevantes, nesta subseção é discutida a análise destes documentos com o intuito de responder as questões de pesquisa deste mapeamento.

Em relação à questão de pesquisa Q1, os artigos de ARENAS e FERNANDEZ (2018), de HÖLBL et al. (2018), de Turkanović et al. (2018), de LIU et al. (2018) e de Han et al. (2018), e as patentes de JASPREET, RANDHAWA (2018), de YUAN, HAIBO (2018), de LIU, NAN; WEI, JINWU (2017) e de DAI, JIANBIAO (2018) trazem sistemas, com propostas similares entre si, para a criação de certificados acadêmicos baseados em *tokens* de *blockchain* a fim de garantir a validade dos certificados, melhorar a segurança dos dados e facilitar a gestão e transferência dos créditos acadêmicos.

Dentre eles, destaca-se o EduCTX (Turkanović et al., 2018), uma plataforma global e descentralizada, baseada em *blockchain*, para gerenciamento de credenciais acadêmicas. Nesta plataforma, temos os *tokens* do EduCTX, que são equivalentes a créditos acadêmicos adquiridos pelo estudante pelas unidades curriculares que ele completou (disciplinas, diplomas, certificados) e que, de forma similar ao Sistema Europeu de Transferência e Acumulação de Créditos (ECTS), busca uma padronização do sistema de créditos para facilitar o processo de transferência de créditos dos alunos entre instituições de ensino superior. O EduCTX é uma *blockchain* federada implementada em Ethereum que utiliza *Delegated Proof-of-Stake* como protocolo de consenso. Dessa forma, apenas os nós das instituições de ensino participam do consenso distribuído.

O artigo de Stefansson e Lentin (2017) apresenta uma criptomoeda (Smileycoin ou SMLY) para recompensar estudantes por seus estudos. O Smileycoin foi idealizado como uma moeda de incentivo para a educação em regiões de baixa renda. Esse artigo descreve a experiência de uso do SMLY como mecanismo de recompensa em um curso de graduação de cálculo, como foi a adoção e uso do SMLY por parte dos alunos e maneiras de responder a usos mal-intencionados do sistema. Destaca-se este trabalho como um dos principais trabalhos relacionados a este TCC pois ele também possui um método de recompensa ao desempenho dos alunos baseado em um ativo digital, neste caso o Smileycoin. O Smileycoin possui um protocolo muito semelhante ao Bitcoin, utilizando *Proof-of-Work* com uma frequência de geração de blocos esperada de 3 minutos. O SMLY pode ser utilizado para cupons que dão ingressos de cinema, voos domésticos ou créditos para celulares.

As patentes SHUIYUAN, XIE et al. (2018) e WANG, CHENGBO; WANG, YITONG (2018) trazem aplicações de *blockchain* para a avaliação de alunos e ajuste da experiência de aprendizado de forma personalizada de acordo com o "ritmo" de aprendizagem de cada aluno, a partir dos resultados anteriores desse aluno armazenados no sistema.

A patente ZHANG, SHENGLI; CHENG, HUAZHENG; WANG, HUI (2018) provê um método de compartilhamento e comercialização de material didático em uma *blockchain*. Autores de material didático submetem o material no sistema por meio de contratos inteligentes e então pessoas podem comprá-los. As características intrínsecas à *blockchain* garantem que os direitos autorais sejam respeitados e os donos sejam devidamente pagos.

Tabela 5 – Características dos principais trabalhos relacionados atualmente.

Nome da aplicação	**Serviço**	**Plataforma**	**Protocolo de Consenso**	**Tipo**
EduCTX	Registro de histórico acadêmico e *token* para créditos acadêmicos	Ethereum	*Delegated-Proof-of-Stake*	Federada
Smileycoin	Criptomoeda	Smileycoin	*Proof-of-Work*	Pública

Fonte: Próprio autor.

Dentre todas as aplicações analisadas para responder à Q1, o EduCTX (HÖLBL et al., 2018) e o Smileycoin (STEFANSSON; LENTIN, 2017) são os principais trabalhos relacionados por suas características supracitadas no texto. A tabela 5 resume seus parâmetros no que se refere às configurações da *blockchain*.

Para responder à questão de pesquisa Q2, foram buscados estudos que ajudem a entender como as propriedades da tecnologia *blockchain* podem ajudar a solucionar problemas da educação ou que tragam propostas de possíveis futuras aplicações desse mecanismo na educação. Os estudos de Jirgensons e Kapenieks (2018), de Duan, Zhong e Liu (2017), de Palma et al. (2019) e de CHENG et al. (2018) apontam a *blockchain* como solução para o problema da falsificação de diplomas e certificados. Ela pode resolver questões de vulnerabilidade, segurança e privacidade no caso de ambientes ubíquos de aprendizagem (BDIWI et al., 2017; BDIWI et al., 2018) e pode ser usada para armazenar registros educacionais relacionados a recompensas de reputação acadêmica (SHARPLES; DOMINGUE, 2016).

Sharples e Domingue (2016) propõem uma solução de *blockchain* que permite usuários assegurarem a confiabilidade de dados acadêmicos baseados na reputação dos criadores destes dados. Assume-se que indivíduos creditam um dado como sendo genuíno se ele foi emitido por uma pessoa ou entidade com uma reputação confiável. Para medir reputação, é proposta uma moeda chamada Kudos. Kudos são distribuídos inicialmente para instituições de renome. Então, uma instituição pode atestar a reputação dos seus funcionários alocando Kudos para eles. Transações envolvendo Kudos são registradas na *blockchain*, assim como os registros acadêmicos.

O estudo de Yakovenko et al. (2019) analisa como as características da tecnologia *blockchain* pode contribuir para instituições de ensino em questões como gerenciamento, redução de custos de energia e de tempo de processamento de informações, e sistemas educacionais. Ele conclui que a transferência de todo o fluxo de documentação de instituições de ensino para a *blockchain* aumentaria a velocidade do processamento dos documentos e também garantiria a transparência e a impossibilidade de perda ou falsificação de documentos. Além disso, facilitaria a transferência de créditos de alunos durante a transição entre instituições.

Hori et al. (2018) apresenta um sistema de ensino que utiliza *e-books*, o CHiLO, e, para resolver os problemas relacionados a direitos autorais do CHiLO, propõe um novo sistema de ensino baseado em *blockchain* e um novo modelo de ensino utilizando moedas virtuais para pagar pelo material didático utilizado.

GONG et al. (2018) propõe uma arquitetura de um sistema de educação abrangente, cobrindo o processo de ensino, avaliação e armazenamento de certificados que combina armazenamento centralizado com armazenamento distribuído em *blockchain*. Esta arquitetura possui 3 camadas: camada de dados, camada lógica e camada de aplicação. Na camada de dados, mecanismos distribuídos para armazenar dados são utilizados para guardar realizações, créditos e prêmios gerados pela educação formal, já os gerados pela educação informal ficam salvos por

um mecanismo dual (parte centralizado e parte distribuído) de armazenamento. Na camada de aplicação, contratos inteligentes são utilizados para "transações" como transferências de créditos e notas entre instituições.

4.6 Considerações do Capítulo

Neste capítulo foi apresentado um mapeamento sistemático para conhecer o estado da arte e da técnica acerca das aplicações de *blockchain* para a educação, com o intuito de responder às questões de pesquisa definidas na subseção 4.2 e poder relacionar o trabalho realizado no TCC com o que já foi desenvolvido na área.

Foram encontrados 108 artigos e 12 patentes na etapa de busca, através da aplicação da *string* de busca definida na subseção 4.3 na base de dados *Scopus* e nas bases de patentes *Espacenet* e INPI. Destes foram selecionados 16 artigos e 7 patentes relevantes, de acordo com os critérios de seleção descritos na subseção 4.4. Em seguida, para cada um dos artigos e patentes selecionados foi elaborada uma descrição acerca do que se trata e dos objetivos dos mesmos.

Constata-se, a partir da visão geral do tópico obtida pelo mapeamento, que este é um tema bastante recente mas que tem despertado grande interesse. Verifica-se que esta tecnologia tem aplicações em diversos setores do sistema educacional, tais como registro de histórico e certificações acadêmicas, método de recompensar os alunos, forma de compartilhamento de material didático garantindo os direitos autorais, sistema de aprendizado personalizado, entre outros.

Nota-se também uma grande diversidade de formas de implementação de soluções baseadas em *blockchain*. São utilizados diferentes plataformas, protocolos de consenso e tipos de *blockchain* em cada trabalho, como evidenciado na Tabela 6, onde estão listados os 10 trabalhos relacionados que ofereceram mais detalhes sobre suas implementações. Dentre as plataformas, Ethereum é a mais comumente utilizada, porém mesmo trabalhos que o utilizam aplicam protocolos de consenso variados. Ademais, pode-se observar diferentes abordagens mesmo para serviços similares.

Por fim, pode-se evidenciar o ineditismo do projeto desenvolvido neste TCC, apesar de se assemelhar a algumas aplicações apresentadas nos quesitos de registro de histórico e de recompensa. Ele é o único trabalho que propõe um *token* utilitário a ser utilizado dentro da relação aluno-professor, com foco a nível de atividades realizadas pelo aluno; ao invés de ter apenas registros a nível de diplomas e certificados, como foi comumente encontrado.

Tabela 6 – Características dos 10 trabalhos relacionados mais descritivos.

Nome da aplicação	**Serviço**	**Plataforma**	**Protocolo de Consenso**	**Tipo**
EduCTX (HÖLBL et al., 2018)	Registro de histórico acadêmico e *token* para créditos acadêmicos	Ethereum	*Delegated-Proof-of-Stake*	Federada
CredenceLedger (ARENAS; FERNANDEZ, 2018)	Registro de histórico acadêmico	Multichain	Protocolo próprio do Multichain	Federada
Blockchain and Smart Contracts for Education Registry (PALMA et al., 2019)	Registro de histórico acadêmico	Ethereum	*Proof-of-Work*	Federada
Kudos (SHARPLES; DOMINGUE, 2016)	Registro de histórico acadêmico e *token* para medir reputação das instituições de ensino superior	Ethereum	*Proof-of-Stake*	Privada
Smileycoin (STEFANSSON; LENTIN, 2017)	Criptomoeda	Smileycoin	*Proof-of-Work*	Pública
Education-Industry Cooperative System (LIU et al., 2018)	Compartilhamento de informações entre universidades e indústrias	Hyperledger Fabric	*Proof-of-Authority*	Federada
Blockchain and Smart Contract for Digital Certificate (CHENG et al., 2018)	Registro de certificados educacionais	Ethereum	Não informa	Federada
Ubiquitous Learning Environment (BDIWI et al., 2018)	Coleta de dados de dispositivos de IoT em salas de aula	Plataforma própria	Protocolo próprio	Privada
CHiLO (HORI et al., 2018)	Sistema de ensino e criptomoeda	Hyperledger Fabric e Bitcoin	*Proof-of-Authority* e *Proof-of-Work*	Pública
Whole-Edu (GONG et al., 2018)	Sistema de ensino	Whole-Edu	Protocolo próprio	Não informa

Fonte: Próprio autor.

5 Resultados e Discussão

Neste capítulo são apresentados os resultados obtidos até o final do projeto. São discutidos alguns fatores que foram relevantes na obtenção desses resultados e suas implicações. O código completo do sistema desenvolvido se encontra no *link*[1] e um manual de utilização do sistema é fornecido no Apêndice A.

5.1 Resultados

O programa entregue atende a todos os requisitos funcionais e não funcionais listados com exceção do RF13, pois não foi possível criar uma rede privada Ethereum e adaptar o projeto para usá-la dentro do prazo de entrega deste trabalho. Sendo assim, as medidas aqui apresentadas se baseiam na rede pública de teste Rinkeby, que foi a utilizada no projeto.

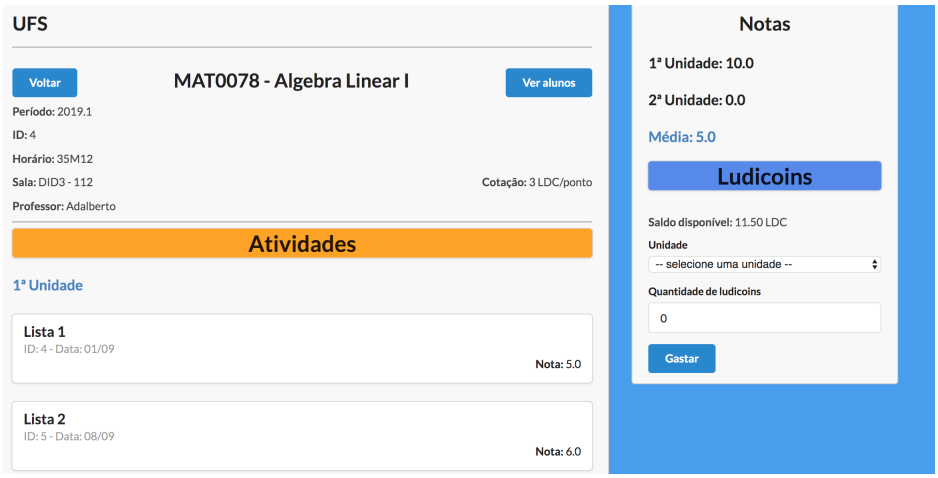
Além disso, também não foi possível dentro do prazo implantar a aplicação *web* em um servidor externo e, por isso, não estão sendo consideradas as possíveis influências que isso acarretaria nos resultados.

Para ilustrar o funcionamento principal do sistema, na Figura 13 tem-se um exemplo de aluno que acumulou, somando os pontos das atividades, mais de 10 pontos em uma unidade e, consequentemente, ganhou Ludicoins. A Figura 14 apresenta o evento de emissão de Ludicoins disparado na blockchain e a Figura 15 mostra uma nota destacada em verde indicando que ela foi incrementada com o uso de Ludicoins.

A seguir serão detalhados os resultados de cada parte do projeto.

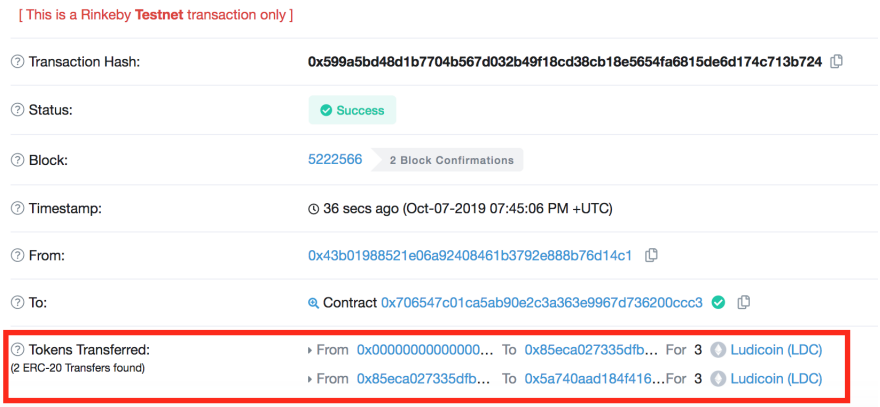
[1] <https://github.com/Walanm/Ludicoin>

Figura 13 – Aluno com ponto sobressalente em uma unidade.



Fonte: Próprio autor.

Figura 14 – Notificação de emissão e concessão de Ludicoins.



Fonte: Próprio autor.

Figura 15 – Nota incrementada por Ludicoins.

Fonte: Próprio autor.

5.1.1 Contratos Inteligentes

Foi realizado um teste de desempenho para avaliar a infra-estrutura utilizada, que é baseada na rede Ethereum pública Rinkeby e também no endereçamento e distribuição de carga de nós feito pelo Infura. O teste consiste em disparar N transações *inserirDisciplina*, do contrato LudiEx, quase simultaneamente e medir quanto tempo leva para uma transação aleatória

disparada ser executada e validada, ou seja, entrar na blockchain dentro de um bloco minerado. A Tabela 7 apresenta os tempos para diferentes valores de N.

Tabela 7 – Tempos para 1 transação aleatória ser validada.

Transações Simultâneas	**Tempo 1**	**Tempo 2**	**Tempo 3**	**Tempo 4**	**Tempo 5**	**Média**
1	24,03s	11,57s	18,19s	21,03s	20,04s	18,97s
10	24,76s	24,78s	14,15s	23,83s	21,65s	21,83s
100	101,60s	113,60s	30,22s	66,07s	79,64s	78,23s
200	135,76s	170,96s	161,54s	138,64s	147,39s	150,86s

Fonte: Próprio autor.

Em adição ao teste de desempenho, foram realizados testes para medir os custos, representados em gas, da implantação dos contratos Ludicoin e LudiEx e das transações executadas pelo contrato LudiEx, que é o único contrato em que o usuário tem acesso direto. Como o custo de uma transação é proporcional ao tamanho das variáveis envolvidas nas operações, foram fixados tamanhos realistas de entradas dos métodos do contrato LudiEx. Para strings, foram utilizadas entradas de 30 caracteres e, para inteiros, foi usado o número 999.999.999.999, que é o maior número de 12 dígitos, valor considerado razoável para entradas como “número de matrícula”, “CPF”, “número de cadastro institucional”; com exceção de entradas onde esse valor estaria muito acima da escala, como “nota” ou “quantidade de unidades”. A Tabela 8 lista os custos de implantação dos contratos e a Tabela 9 lista os custos encontrados das transações.

Tabela 8 – Custos para implantação dos contratos.

Contrato	**Custo (em ether)**
Ludicoin	0,001585
LudiEx	0,006665

Fonte: Próprio autor.

Uma observação importante em relação aos contratos inteligentes confeccionados é o fato de que o contrato LudiEx possui um tamanho muito próximo do tamanho máximo para contratos (24576 *bytes*) do Ethereum, o que impede futuros incrementos sem que haja uma refatoração que distribua para outros contratos algumas funções desempenhadas por ele.

5.1.2 Aplicação Web

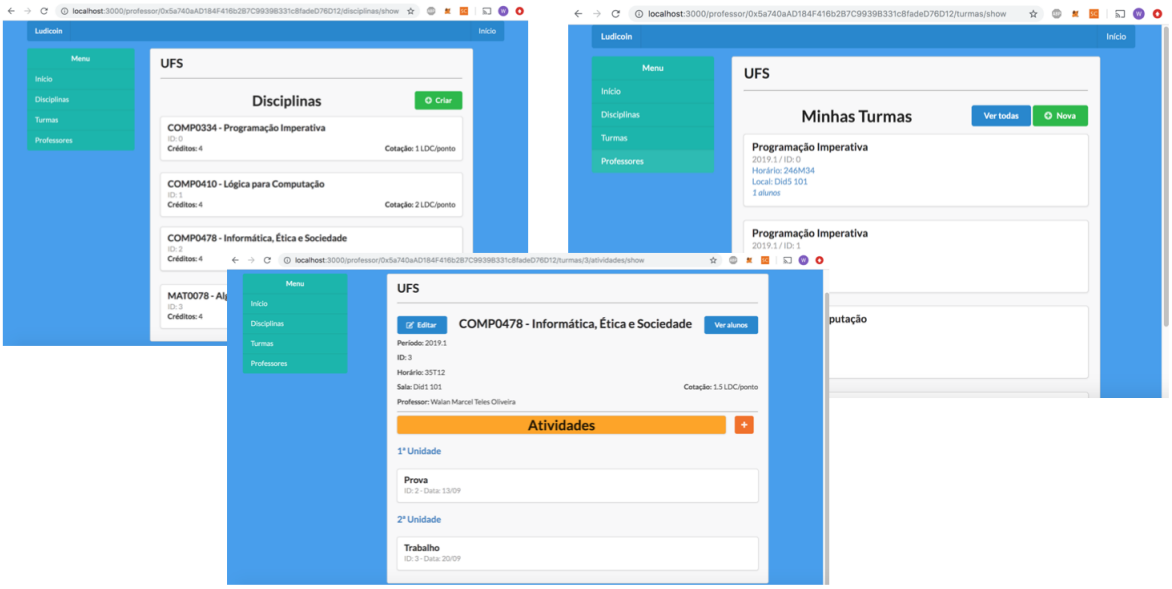
Ao todo foram criadas 24 telas para realizar todas as tarefas definidas nos casos de uso. No *link*[2] encontram-se todas as telas criadas. A Figura 16 possui exemplos de telas com listas de dados e a Figura 17 possui exemplos de telas de cadastro de dados. Já a Figura 18 apresenta os 3 principais tipos de telas com os quais os alunos interagem dentro do sistema.

Tabela 9 – Custos das transações de LudiEx.

Método	Custo (em ether)	Método	Custo (em ether)
aceitarProfessor	0,000198	gastarLudicoins	0,000049
atualizarAluno	0,000071	inicializar	0,000158
atualizarDisciplina	0,000132	inserirDisciplina	0,000263
atualizarProfessor	0,000073	inserirUnidade	0,000079
atualizarTurma	0,000086	matricularNaTurma	0,000108
cadastrarAluno	0,000233	removerDisciplina	0,000059
cadastrarAtividade	0,000249	removerUnidade	0,000034
cadastrarAtividadeRealizada	0,000068	requisitarCadastroProfessor	0.000233
cadastrarTurma	0,000938	requisitarMatriculaNaTurma	0.000064

Fonte: Próprio autor.

Figura 16 – Telas com Listas de Dados.



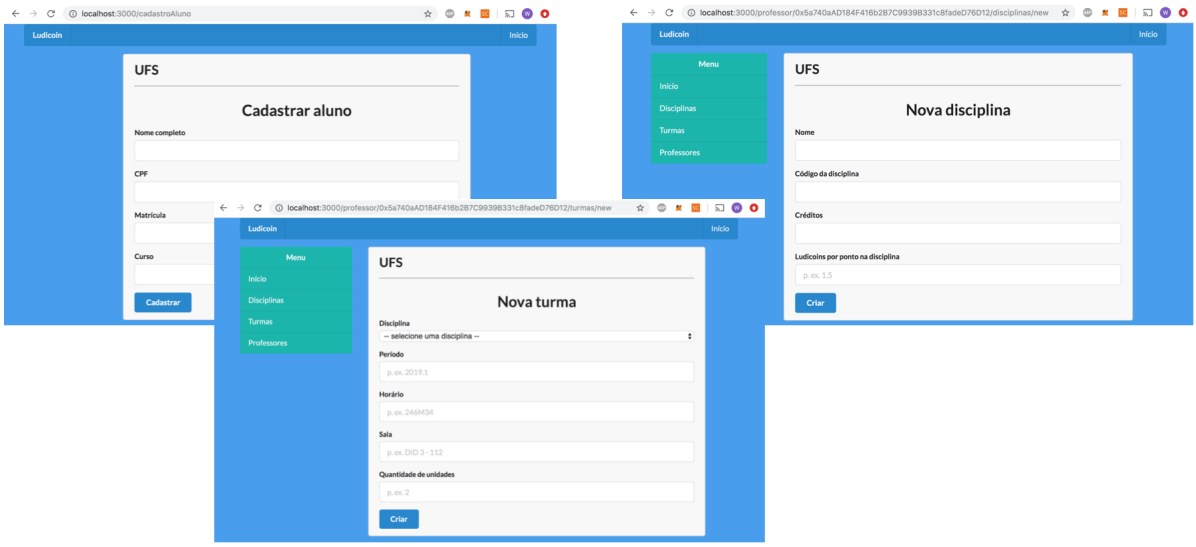
Fonte: Próprio autor.

Na Figura 19 é mostrado um exemplo de como o layout responsivo se adapta ao tamanho de tela, atendendo ao requisito não-funcional RNF1. Na Figura 20 é mostrado um exemplo de mensagem de erro, atendendo ao requisito não-funcional RNF3.

A aplicação também foi testada em todos os navegadores compatíveis com o MetaMask (Chrome, Firefox, Opera e Brave) e funcionou em todos eles, atendendo ao requisito não-funcional RNF2. A aplicação *web* está hospedada em *localhost*.

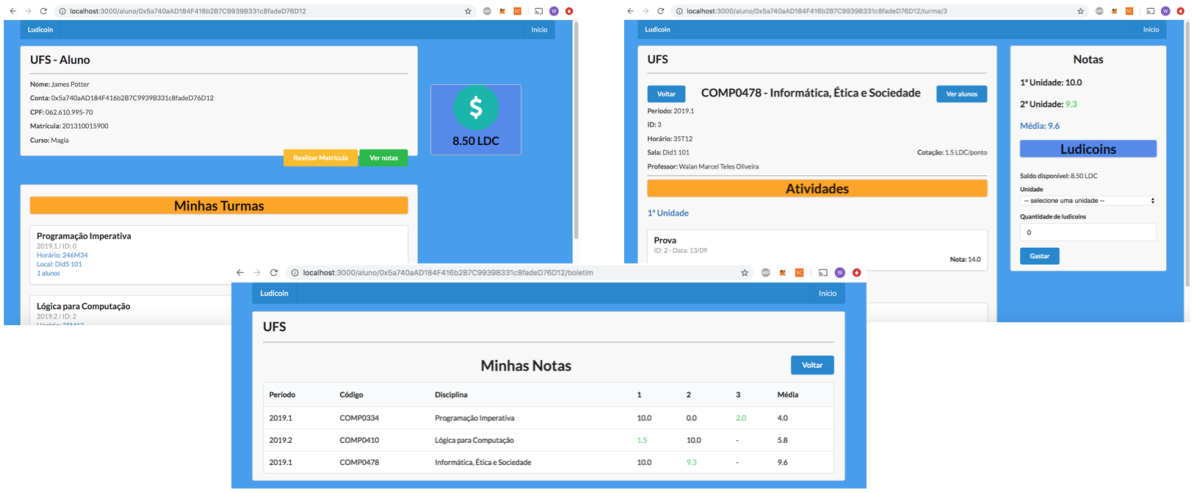
[2] <https://github.com/Walanm/Ludicoin/tree/master/Telas>

Figura 17 – Telas de Cadastro de Dados.



Fonte: Próprio autor.

Figura 18 – Telas de Interação dos Alunos.



Fonte: Próprio autor.

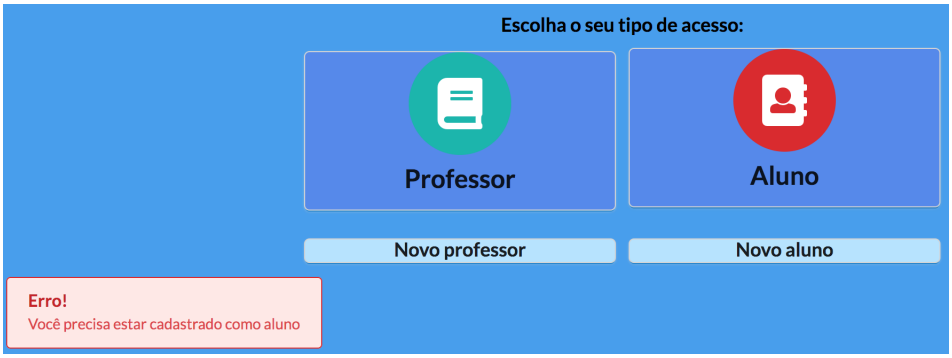
5.1.3 Integração da Blockchain com a Aplicação Web

As páginas *web* conseguem acessar informações e enviar transações para a *blockchain*. Ao enviar uma transação, é disparada uma confirmação pelo MetaMask, como pode-se ver na Figura 21. Enquanto a transação espera ser validada pela *blockchain*, a página ativa uma animação de carregamento no botão que disparou a transação, como pode ser visto na Figura 22.

Figura 19 – Responsividade do Layout.

Fonte: Próprio autor.

Figura 20 – Exemplo de Mensagem de Erro do Sistema.



Fonte: Próprio autor.

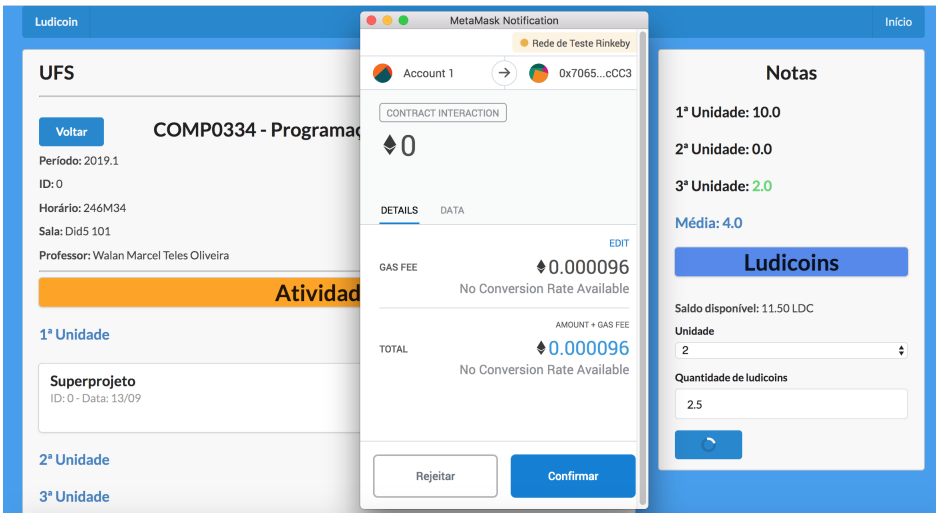
5.1.4 Validação do Modelo de Negócio

O questionário foi respondido por 79 participantes. Destes 76 foram alunos e apenas 3 foram professores. A Figura 23 apresenta o perfil acadêmico dos participantes. As respostas em relação às afirmações foram dadas de acordo com a escala Likert. Como exposto anteriormente, as 3 assertivas avaliadas foram:

Q1) Guardar pontos extras para usar em outras unidades ou disciplinas pode ser útil para motivar os alunos a realizarem todas as suas atividades.

Q2) "Ludicoins" devem ser gerados somente quando o aluno atingir pontuação maior que 10

Figura 21 – Submissão de Transação.



Fonte: Próprio autor.

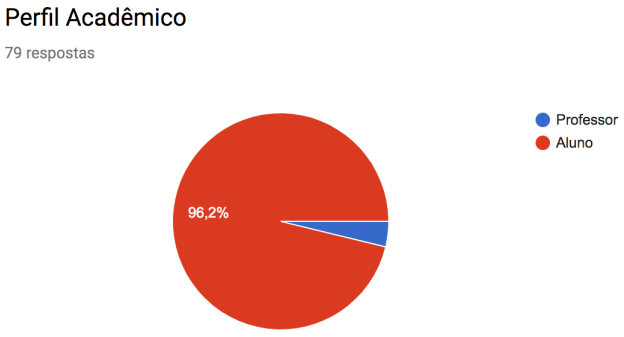
Figura 22 – Indicador de Carregamento.

Fonte: Próprio autor.

em uma unidade de uma disciplina.

Q3) Pontos de diferentes disciplinas devem ter diferentes valores quando convertidos para "Ludicoins".

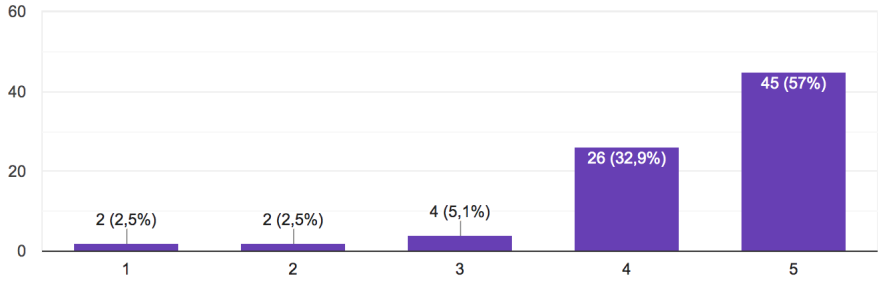
Figura 23 – Perfil Acadêmico dos Participantes.



Fonte: Próprio autor.

Em relação ao item Q1, uma ampla maioria (89,9%) concordou com a hipótese levantada na proposta de negócio. A Figura 24 apresenta o gráfico em coluna das respostas a este item.

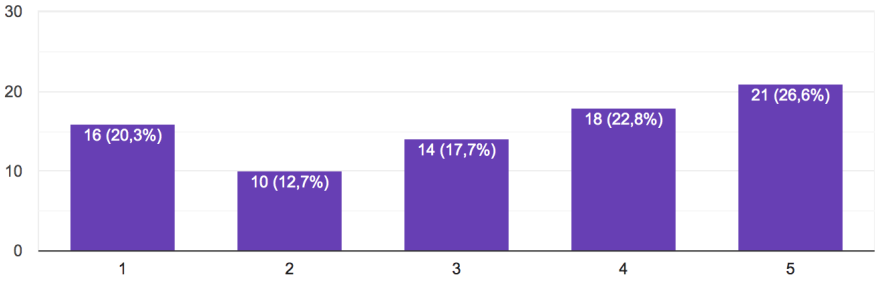
Figura 24 – Respostas Referentes ao Item Q1.



Fonte: Próprio autor.

Já em relação ao item Q2, apenas 49,4% dos participantes concordaram com a regra de negócio que restringe o ganho de Ludicoins a quando um aluno atinge uma nota maior que 10 em uma unidade de uma disciplina. A Figura 25 apresenta as respostas a este item.

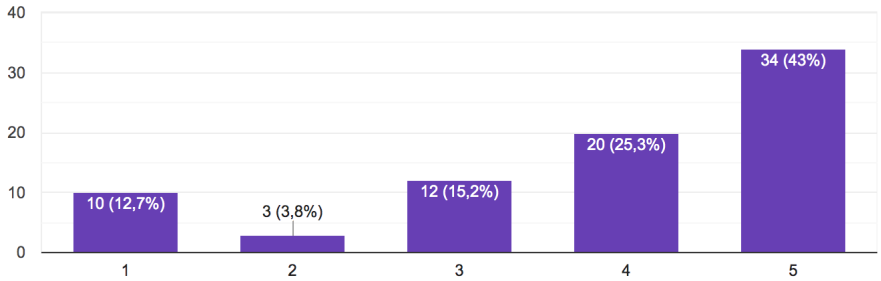
Figura 25 – Respostas Referentes ao Item Q2.



Fonte: Próprio autor.

Em relação ao item Q3, a maioria (68,3%) concordou com diferentes equivalências, ou taxas de câmbio, entre pontos nas disciplinas e Ludicoins. A Figura 26 apresenta as respostas a esse item.

Figura 26 – Respostas Referentes ao Item Q3.



Fonte: Próprio autor.

No espaço para comentários e sugestões, 23 participantes deixaram respostas. Dentre elas, destacam-se sugestões sobre outras modalidades de obtenção de Ludicoins, com 7 sugestões sobre esse tema, e sugestões sobre como seria feita a conversão de pontos em Ludicoins, com 4 sugestões sobre esse tema. Todos os comentários encontram-se reproduzidos no Apêndice B.

Sobre o levantamento feito acerca de alunos que poderiam ser beneficiados pela implantação do projeto no ano-período 2019.1 da UFS, dos 12 professores consultados, apenas 5 responderam. Destes, foram identificados apenas 3 alunos que se encaixam no perfil procurado. Esses 5 professores somados lecionam para um total de 11 turmas e 441 alunos, podendo haver interseções entre as turmas.

5.2 Discussão

Do ponto de vista funcional, os requisitos foram atendidos de forma que permitam a realização da proposta principal do sistema. Todos os requisitos foram atendidos com exceção do requisito funcional RF13, pois não foi utilizar uma rede de *blockchain* privada dentro do prazo. O *token* universitário Ludicoin pode ser considerado um token utilitário, pois é um *token* que permite o uso de uma funcionalidade, que é ganhar pontos numa unidade de uma disciplina ao gastar Ludicoins. O uso da tecnologia *blockchain* garante que não haverá gasto duplo, ou seja, um mesmo Ludicoin ser utilizado duas vezes; e que não haverá adulterações nos dados, que têm a consistência garantida pela estrutura dos blocos que compõem a cadeia.

Quanto ao uso da *blockchain*, a implantação do sistema em uma rede Ethereum pública de testes tem caráter provisório para fins de desenvolvimento, pois as redes de teste possuem uma instabilidade inerente devido a servirem como experimentação de variações do protocolo Ethereum. Para uma implementação definitiva do sistema, deve-se considerar o uso da rede pública principal ou de uma rede privada, que foi a originalmente prevista para esse trabalho.

A rede Ethereum pública de testes Rinkeby tem o tempo para a criação de bloco fixado em 15 segundos e, como os contratos estão rodando em cima de uma rede pública, as transações do Ludicoin estão concorrendo com as transações de outros contratos da rede para serem escolhidas na mineração de blocos. Considerando também o tempo de transmissão de uma transação para os nós da *blockchain*, o tempo médio de validação de uma transação (18,97s) encontra-se dentro do esperado. Há uma variação de tempo significativa para tráfegos intensos, chegando a tempos médios de 79,64s e 147,38s para 100 e 200 transações simultâneas, respectivamente.

Em uma solução com rede privada, esses tempos poderiam ser reduzidos. Atualmente, o tamanho máximo de um bloco Ethereum comporta até 10.000.000 de *gas*, o que equivale a 0,01 *ether* (ETHEREUM, 2019a). Como a transação testada *inserirDisciplina* custa 0,000263 *ether*, caberiam até 38 transações desse tipo por bloco, o que validaria 100 transações em 3 blocos e 200 transações em 6 blocos. Fixando a taxa de geração de blocos em 15s, seriam validadas todas as transações simultâneas em aproximadamente 45s e 90s, respectivamente, desconsiderando os

tempos de transmissão e possíveis congestionamentos na rede.

Em uma rede de testes como a Rinkeby, o custos das transações são simbólicos, visto que o usuário pode solicitar mais *ethers* a distribuidores de *ether* na rede. Não obstante, em uma implementação em rede privada, tal qual foi inicialmente planejada para este trabalho, esses custos podem servir de incentivo para os usuários manterem nós na rede para minerarem blocos e receberem *ethers*, a partir do protocolo de consenso *Proof-of-Work*. Como a quantidade de mineradores seria restrita, devido ao caráter privado da rede, alunos e professores poderiam facilmente acumular *ethers* o suficiente para usar o sistema. A aplicação web produzida pode ser facilmente adaptada para interagir com uma rede privada.

Redes Ethereum que utilizam o consenso *Proof-of-Authority* não recompensam os participantes com *ether*, então outros incentivos teriam que ser considerados para redes privadas que o utilizassem. Neste caso, uma forma de incentivo poderia ser apenas permitir o acesso ao sistema se a máquina do usuário tivesse um nó ativo localmente. Uma outra solução poderia ser a implementação de um cliente Ethereum de baixo custo computacional a ser carregado pela página *web* para contribuir com a *blockchain* enquanto o usuário permanecer nas páginas *web* do Ludicoin. Esse cliente receberia a permissão de nó validador de forma automatizada por algum nó validador já presente na rede.

Do ponto de vista usabilidade, o elevado tempo de espera (maior que 10s) ao se submeter uma transação pode ser considerado um problema. As páginas *web* da aplicação possuem indicadores de carregamento, porém, mesmo assim, usuários podem querer realizar outras tarefas e se frustrarem com a espera. O fato do tempo de espera poder variar muito também pode ser considerada uma desvantagem do sistema. Como não foi possível hospedar a aplicação *web* em um servidor remoto dentro do prazo, não foram medidas as possíveis influências que isso poderia ter no carregamento de páginas.

Os resultados do questionário aplicado apontam que a ampla maioria dos participantes concordam com a hipótese que sustenta a proposta do projeto e que a maioria aceita a regra de diferentes equivalências entre Ludicoins e pontos em diferentes disciplinas. No entanto, a regra de atribuição de Ludicoins não teve a aceitação da maioria, possivelmente por ser muito restritiva. Essa hipótese é corroborada pelo número reduzido de alunos encontrados que poderiam ser beneficiados pela aplicação do Ludicoin no ano-período 2019.1 da UFS. Sendo assim, é importante considerar as regras de negócio atuais como um ponto de partida e novas regras podem ser incorporadas para que Ludicoins sejam atribuídos de forma satisfatória. Dentre as sugestões que se destacam estão a remuneração de Ludicoins diretamente por realização de atividades extra-classe, por assiduidade, por realização de atividades específicas e por monitorias e estágios.

Os principais fatores limitantes do projeto foram os atrasos no desenvolvimento devido a refatorações feitas no contrato LudiEx nas vezes em que o tamanho máximo do contrato foi excedido e devido ao tempo maior que o esperado no desenvolvimento do *front-end* da aplicação.

6

Conclusão e Trabalhos Futuros

Neste trabalho foi desenvolvido um *token* universitário, chamado de Ludicoin, por meio da *blockchain* Ethereum e uma aplicação web que permite a sua utilização por alunos e professores. Ludicoins são concedidos a alunos universitários que possuam pontos sobressalentes - acima da pontuação máxima 10 - nas disciplinas acadêmicas. Também foi feito um mapeamento sistemático das pesquisas e projetos de *blockchain* aplicados em educação. Ademais, foi aplicado um questionário a alunos e professores universitários para a validação da proposta e das regras de negócio do projeto.

Pode-se considerar que o projeto foi bem-sucedido em confeccionar o software do sistema, porém não foi obtido êxito em criar uma infra-estrutura própria para o uso deste, pois não não houve tempo dentro do prazo deste trabalho para a realização de tal atividade. Ao invés disso, foi feito uso de uma rede pública de *blockchain*. O software consiste em contratos inteligentes implantados na *blockchain* Ethereum e uma aplicação web que interage com esses contratos. Esses contratos podem ser implantados em qualquer rede Ethereum e a aplicação web pode ser facilmente adaptada para interagir com uma nova rede escolhida.

Os resultados do questionário indicam que há grande aceitação da proposta do projeto. Não obstante, as regras de negócio ainda precisam ser refinadas de modo que sejam satisfatórias para alunos e professores e possam causar um impacto maior no desempenho dos estudantes.

Por fim, deve-se considerar algumas limitações do trabalho realizado. Tanto o questionário quanto o levantamento realizados tiveram um alcance aquém do desejado e, portanto, seus resultados, apesar de conterem informações valiosas, não devem ser considerados conclusivos. Além disso, o projeto não foi implantado em uma rede privada como o planejado e, por isso, não foram feitas medições com relação aos aspectos de redes privadas.

Como sugestão para trabalhos futuros, pode ser considerada a aplicação do Ludicoin em turmas universitárias e uma nova etapa de validação dentre os usuários. Também poderá ser feita a implantação de uma rede privada e, com isso, poderão ser feitas medições de desempenho

da rede. Nessa implantação, deverão ser resolvidas questões a respeito dos incentivos para participantes da rede, do protocolo de consenso a ser utilizado e da comunicação entre os nós, sendo recomendada a criação de um ou mais nós na nuvem para servir como ponto de encontro. Finalmente, a implementação de novas regras para a concessão de Ludicoins formuladas a partir das informações obtidas pelo questionário podem apontar caminhos para uma aplicação mais satisfatória do Ludicoin.

Referências

ANTONOPOULOS, A. M. *Mastering Bitcoin: Programming the open blockchain*. [S.l.]: "O'Reilly Media, Inc.", 2017. Citado na página 21.

ARENAS, R.; FERNANDEZ, P. Credenceledger: A permissioned blockchain for verifiable academic credentials. In: *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. [S.l.: s.n.], 2018. p. 1–6. Citado 2 vezes nas páginas 49 e 53.

BARRETO, H. V. D. *Metodologia de Gestão Ludus*. 2019. Disponível em: <https://github.com/hugodbarreto/ludus>. Acesso em: 23 out 2019. Citado na página 31.

BASHIR, I. *Mastering blockchain*. 1a. ed. [S.l.]: Packt Publishing Ltd, 2017. ISBN 9781787125445. Citado 9 vezes nas páginas 13, 16, 17, 19, 21, 22, 24, 25 e 29.

BDIWI, R. et al. Towards a new ubiquitous learning environment based on blockchain technology. In: *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*. [S.l.: s.n.], 2017. p. 101–102. ISSN 2161-377X. Citado na página 51.

BDIWI, R. et al. A blockchain based decentralized platform for ubiquitous learning environment. In: *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*. [S.l.: s.n.], 2018. p. 90–92. ISSN 2161-377X. Citado 2 vezes nas páginas 51 e 53.

BENTOV, I. et al. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 42, n. 3, p. 34–37, dez. 2014. ISSN 0163-5999. Disponível em: <http://doi.acm.org/10.1145/2695533.2695545>. Citado na página 24.

BUTERIN, V. et al. A next-generation smart contract and decentralized application platform. 2014. Citado 2 vezes nas páginas 26 e 28.

CARVER, J.; MERRIAM, P. *eth-hash*. 2018. Disponível em: <https://github.com/ethereum/eth-hash>. Acesso em: 7 out 2019. Citado na página 38.

CASTRO, M.; LISKOV, B. Practical byzantine fault tolerance. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 1999. (OSDI '99), p. 173–186. ISBN 1-880446-39-1. Disponível em: <http://dl.acm.org/citation.cfm?id=296806.296824>. Citado na página 22.

CHENG, J. et al. Blockchain and smart contract for digital certificate. In: *2018 IEEE International Conference on Applied System Invention (ICASI)*. [S.l.: s.n.], 2018. p. 1046–1051. Citado 2 vezes nas páginas 51 e 53.

CHOHAN, U. W. The double spending problem and cryptocurrencies. *Available at SSRN 3090174*, 2017. Citado na página 21.

Christidis, K.; Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *IEEE Access*, v. 4, p. 2292–2303, 2016. ISSN 2169-3536. Citado 3 vezes nas páginas 13, 17 e 25.

COULOURIS, G. et al. *Distributed Systems: Concepts and Design*. 5th. ed. USA: Addison-Wesley Publishing Company, 2011. ISBN 0132143011, 9780132143011. Citado na página 19.

DAI, JIANBIAO. *Digital learning experience management method based on blockchain token technology*. SHANGHAI NETBAN EDUCATION TECH COMPANY LIMITED. China CN108768614. Novembro, 2018. Citado na página 49.

DALMORO, M.; VIEIRA, K. M. Dilemas na construção de escalas tipo likert: o número de itens e a disposição influenciam nos resultados? *Revista gestão organizacional*, v. 6, n. 3, 2014. Citado na página 32.

DENNY, P. The effect of virtual achievements on student engagement. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2013. (CHI '13), p. 763–772. ISBN 978-1-4503-1899-0. Disponível em: <http://doi.acm.org/10.1145/2470654.2470763>. Citado na página 27.

Duan, B.; Zhong, Y.; Liu, D. Education application of blockchain technology: Learning outcome and meta-diploma. In: *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*. [S.l.: s.n.], 2017. p. 814–817. ISSN 1521-9097. Citado na página 51.

ETHEREUM. *Learn about Ethereum*. 2019. Disponível em: <http://ethereum.org/learn/>. Acesso em: 7 out 2019. Citado 2 vezes nas páginas 38 e 62.

ETHEREUM. *web3.js Documentation*. 2019. Disponível em: <https://web3js.readthedocs.io/en/v1.2.1/>. Acesso em: 7 out 2019. Citado na página 37.

FARDO, M. L. A gamificação aplicada em ambientes de aprendizagem. *RENOTE*, v. 11, n. 1, 2013. Citado na página 27.

FILHO, J. R. F.; BRAGA, A. M.; LEAL, R. L. V. *Tecnologia Blockchain: uma visão geral*. 2016. Citado na página 28.

GONG, X. et al. Parallel-education-blockchain driven smart education: Challenges and issues. In: *2018 Chinese Automation Congress (CAC)*. [S.l.: s.n.], 2018. p. 2390–2395. Citado 2 vezes nas páginas 51 e 53.

GRECH, A.; CAMILLERI, A. F. *Blockchain in education*. [S.l.]: Luxembourg: Publications Office of the European Union, 2017. Citado na página 13.

GREVE, F. et al. Blockchain e a revolução do consenso sob demanda. *Livro de Minicursos do SBRC*, v. 1, p. 1–52, 2018. Citado 3 vezes nas páginas 17, 19 e 20.

HABER, S.; STORNETTA, W. S. How to time-stamp a digital document. *J. Cryptol.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 3, n. 2, p. 99–111, jan. 1991. ISSN 0933-2790. Disponível em: <http://dx.doi.org/10.1007/BF00196791>. Citado na página 16.

HAMARI, J. et al. Does gamification work?-a literature review of empirical studies on gamification. In: *HICSS*. [S.l.: s.n.], 2014. v. 14, n. 2014, p. 3025–3034. Citado na página 27.

HAN, M. et al. A novel blockchain-based education records verification solution. In: . [S.l.: s.n.], 2018. p. 178–183. Citado na página 49.

HOFFSTEIN, J. et al. *An introduction to mathematical cryptography*. [S.l.]: Springer, 2008. v. 1. Citado na página 18.

HORI, M. et al. Learning system based on decentralized learning model using blockchain and sns. In: . [S.l.: s.n.], 2018. p. 183–190. Citado 2 vezes nas páginas 51 e 53.

HOUBEN, R.; SNYERS, A. Cryptocurrencies and blockchain: legal context and implications for financial crime, money laundering and tax evasion. 2018. Citado 2 vezes nas páginas 23 e 26.

HÖLBL, M. et al. Eductx: An ecosystem for managing digital micro-credentials. In: *2018 28th EAEEIE Annual Conference (EAEEIE)*. [S.l.: s.n.], 2018. p. 1–9. ISSN 2472-7687. Citado 4 vezes nas páginas 25, 49, 51 e 53.

INC., F. *Documentos*. 2019. Disponível em: <https://pt-br.reactjs.org/docs/getting-started.html>. Acesso em: 7 out 2019. Citado na página 29.

INFURA. *Infura Documentation*. 2018. Disponível em: <https://infura.io/docs>. Acesso em: 7 out 2019. Citado na página 30.

JASPREET, RANDHAWA. *Methods and systems for employment and education verification using blockchain*. Estados Unidos US2018293547. Novembro, 2018. Citado na página 49.

JIRGENSONS, M.; KAPENIEKS, J. Blockchain and the future of digital learning credential assessment and management. *Journal of Teacher Education for Sustainability*, v. 20, n. 1, p. 145–156, 2018. Disponível em: <https://content.sciendo.com/view/journals/jtes/20/1/article-p145.xml>. Citado na página 51.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007. Citado na página 46.

KRUCHTEN, P. The 4+1 view model of architecture. *IEEE Software*, v. 12, p. 45–50, 11 1995. Citado na página 39.

LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, ACM, New York, NY, USA, v. 4, n. 3, p. 382–401, jul. 1982. ISSN 0164-0925. Disponível em: <http://doi.acm.org/10.1145/357172.357176>. Citado na página 21.

LANSKY, J. Possible state approaches to cryptocurrencies. *Journal of Systems Integration*, v. 8, 01 2018. Citado na página 25.

LIKERT, R. A technique for the measurement of attitudes. *Archives of psychology*, 1932. Citado 2 vezes nas páginas 14 e 32.

LIU, NAN; WEI, JINWU. *Education certification method and system based on blockchain*. CHINA UNICOM. China CN107483498. Dezembro, 2017. Citado na página 49.

LIU, Q. et al. Education-industry cooperative system based on blockchain. In: *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. [S.l.: s.n.], 2018. p. 207–211. Citado 2 vezes nas páginas 49 e 53.

METAMASK. *MetaMask Developer Documentation*. 2019. Disponível em: <https://metamask.github.io/metamask-docs/>. Acesso em: 7 out 2019. Citado na página 29.

NAKAMOTO, S. *Bitcoin: A peer-to-peer electronic cash system*. 2008. Disponível em: <http://www.bitcoin.org/bitcoin.pdf>. Citado 5 vezes nas páginas 13, 16, 21, 22 e 23.

NARAYANAN, A. et al. *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. [S.l.]: Princeton University Press, 2016. Citado na página 20.

PALMA, L. M. et al. Blockchain and smart contracts for higher education registry in brazil. *International Journal of Network Management*, v. 0, n. 0, p. e2061, 2019. E2061 nem.2061. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2061>. Citado 2 vezes nas páginas 51 e 53.

PENARD, W.; WERKHOVEN, T. v. *On the Secure Hash Algorithm family*. 2008. Disponível em: <https://www.staff.science.uu.nl/~tel00101/liter/Books/CrypCont.pdf>. Citado na página 18.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. Swindon, UK: BCS Learning & Development Ltd., 2008. (EASE'08), p. 68–77. Disponível em: <http://dl.acm.org/citation.cfm?id=2227115.2227123>. Citado na página 46.

RIES, E. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, 2011. ISBN 9780307887894. Disponível em: <https://books.google.com.br/books?id=r9x-OXdzpPcC>. Citado na página 31.

ROHR, J.; WRIGHT, A. Blockchain-based token sales, initial coin offerings, and the democratization of public capital markets. *Hastings LJ*, HeinOnline, v. 70, p. 463, 2018. Citado na página 26.

SHARPLES, M.; DOMINGUE, J. The blockchain and kudos: A distributed system for educational record, reputation and reward. In: . [S.l.: s.n.], 2016. v. 9891, p. 490–496. ISBN 978-3-319-45152-7. Citado 2 vezes nas páginas 51 e 53.

SHUIYUAN, XIE et al. *System and method of evaluating comprehensive quality of students, based on block chain technology*. GUANGZHOU YANGGU SOFTWARE CO LTD. China CN108764686. Novembro, 2018. Citado na página 50.

STEFANSSON, G.; LENTIN, J. From smileys to smileycoins: Using a cryptocurrency in education. v. 2, 01 2017. Citado 3 vezes nas páginas 50, 51 e 53.

SZABO, N. Formalizing and securing relationships on public networks. *First Monday*, v. 2, n. 9, 1997. Citado na página 25.

TANENBAUM, A. *Redes de computadores*. [S.l.]: CAMPUS - RJ, 2003. ISBN 9788535211856. Citado na página 19.

TANENBAUM, A. S.; STEEN, M. v. *Distributed Systems: Principles and Paradigms (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 0132392275. Citado na página 19.

TAYLOR, P. J. et al. A systematic literature review of blockchain cyber security. *Digital Communications and Networks*, 2019. ISSN 2352-8648. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2352864818301536>. Citado na página 13.

Turkanović, M. et al. Eductx: A blockchain-based higher education credit platform. *IEEE Access*, v. 6, p. 5112–5127, 2018. ISSN 2169-3536. Citado 2 vezes nas páginas 49 e 50.

VOGELSTELLER, F.; BUTERIN, V. *EIP 20: ERC-20 Token Standard*. 2019. Disponível em: <https://eips.ethereum.org/EIPS/eip-20>. Acesso em: 7 out 2019. Citado na página 39.

WANG, CHENGBO; WANG, YITONG. *Educational evaluation system based on blockchain technology and terminal thereof*. China CN108682195. Outubro, 2018. Citado na página 50.

WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, v. 151, p. 1–32, 2014. Citado na página 26.

YAKOVENKO, I. et al. The blockchain technology as a catalyst for digital transformation of education. *International Journal of Mechanical Engineering and Technology*, n. 1, p. 886–897, 2019. Cited By 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85060918729&partnerID=40&md5=da39e52ad1c377ba6518e1ca4a85d7dd>. Citado na página 51.

YUAN, HAIBO. *Education degree issuing method and system based on blockchain*. SICHUAN CHANGHONG ELECTRIC CO LTD. China CN107977910. Maio, 2018. Citado na página 49.

ZHANG, SHENGLI; CHENG, HUAZHENG; WANG, HUI. *Education resource sharing method and system based on block chain*. UNIV SHENZHEN. China CN108734576. Novembro, 2018. Citado na página 50.

Apêndices

APÊNDICE A – Manual de Uso do Ludicoin

É apresentado aqui um manual para a utilização do sistema Ludicoin. Caso haja quaisquer dúvidas, entre em contato pelo *email* walan.dn@gmail.com.

O Ludicoin é um sistema implementado na plataforma de *blockchain* Ethereum com o objetivo de permitir o acúmulo de pontos sobressalentes nas disciplinas da universidade na forma de *tokens*, os "Ludicoins", para serem usados por alunos e professores do ambiente acadêmico. Dentro deste sistema o aluno pode guardar pontos extras que sobrarem em unidades das disciplinas e utilizá-los em unidades posteriores ou em outras disciplinas de professores participantes da rede.

Um aluno será premiado com Ludicoins quando a soma das notas recebidas por prova e atividades, submetidas no sistema por um professor, de uma mesma disciplina, numa mesma turma e numa mesma unidade excederem a nota máxima 10. Depois disso, os alunos poderão utilizar os *tokens* acumulados em transações por notas em outras matérias. Cada disciplina tem uma equivalência diferente entre pontos e Ludicoins.

A.1 Pré-requisitos

Alguns programas e preparações são necessários para poder executar o programa. Para atender aos pré-requisitos, faça como segue:

1) Instale o Node.js e o gerenciador de pacotes NPM:

a) No Windows: acesse <https://nodejs.org/en/download/> e baixe o instalador. Execute o arquivo baixado e faça a instalação padrão. Ela instalará o Node.js e o NPM;

b) No Linux: execute os seguintes comandos no Terminal para instalar ambos:

- sudo apt-get update
- sudo apt-get install nodejs

2) Instale o Git:

a) No Windows: acesse <https://gitforwindows.org/> e baixe o instalador. Execute o arquivo baixado;

b) No Linux: execute os seguintes comandos no Terminal:

- sudo apt-get update
- sudo apt-get install git

3) Configure seu nome de usuário e email no Git executando os seguintes comandos no Terminal ou CMD do Windows (substitua as palavras entre aspas pelos seus dados):

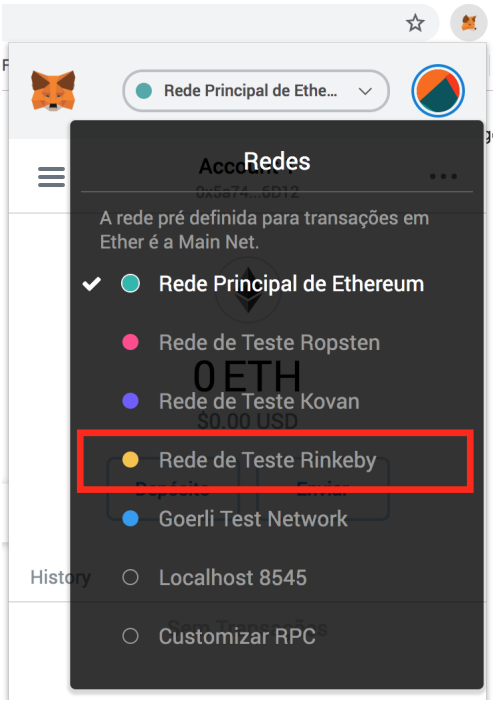
- git config –global user.name "Seu nome"
- git config –global user.email "seu@email.com"

4) Instale a extensão MetaMask no seu navegador: acesse <https://metamask.io/> e selecione o seu navegador para realizar a instalação. Siga os passos sugeridos para criar a sua conta no MetaMask;

5) Clique no ícone do MetaMask no seu navegador e entre com a sua senha para acessar sua conta;

6) Clique em "Rede Principal de Ethereum" (ou *Main Ethereum Network*, em inglês) no MetaMask e selecione a opção "Rede de Teste Rinkeby" (ou *Rinkeby Test Network*, em inglês). A figura 27 ilustra esse passo;

Figura 27 – Seleção da rede Rinkeby no MetaMask.



Fonte: Próprio autor.

7) Receba *ethers* da rede Rinkeby para poder utilizar o Ludicoin:

- Clique no ícone do MetaMask e em seguida clique em "Conta 1"(ou "Account 1") para copiar o endereço da sua conta.

- Acesse <http://rinkeby-faucet.com>, cole o endereço da conta e submeta. Espere alguns segundos e você receberá 0,001 *ether* na sua conta;
- Ou faça uma publicação no Facebook ou Twitter informando o endereço da sua conta, acesse <https://www.rinkeby.io/#faucet> e cole o endereço url da publicação. Dessa forma você pode receber até 18.75 *ethers* na sua conta (escolha a quantidade entre as opções);

A.2 Execução da aplicação

Para executar a aplicação web, execute os seguintes comandos no Terminal do Linux ou no CMD do Windows:

- git clone https://github.com/Walanm/Ludicoin.git
- cd Ludicoin
- npm run dev

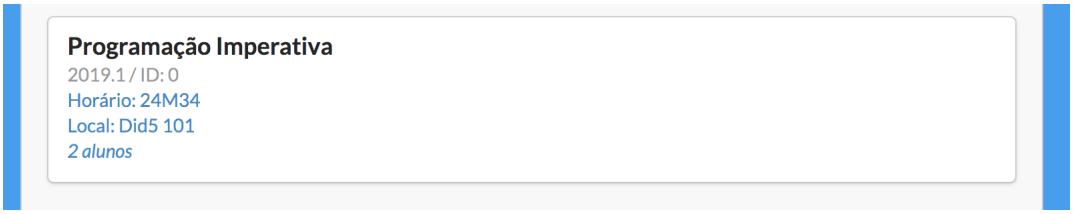
Em seguida abra o navegador no endereço <localhost:3000>. Você deverá estar logado na sua conta no MetaMask. Ao entrar em alguma das opções da página inicial surgirá um *pop-up* pedindo para conectar o MetaMask ao site. Clique em "OK"para aceitar.

A.3 Orientações para uso da aplicação

Essa seção contém orientações gerais para o uso da aplicação. É necessário estar logado no MetaMask dentro do seu navegador de uso para poder realizar ações no *site*.

A navegação pelo site é feita a partir de elementos como botões, cartões e menus. Cartões apresentam informações sobre alguma entidade (por exemplo, turmas ou disciplinas), e ao ser clicado, direcionará o usuário para uma página com os detalhes daquela entidade. A Figura 28 apresenta um exemplo de cartão.

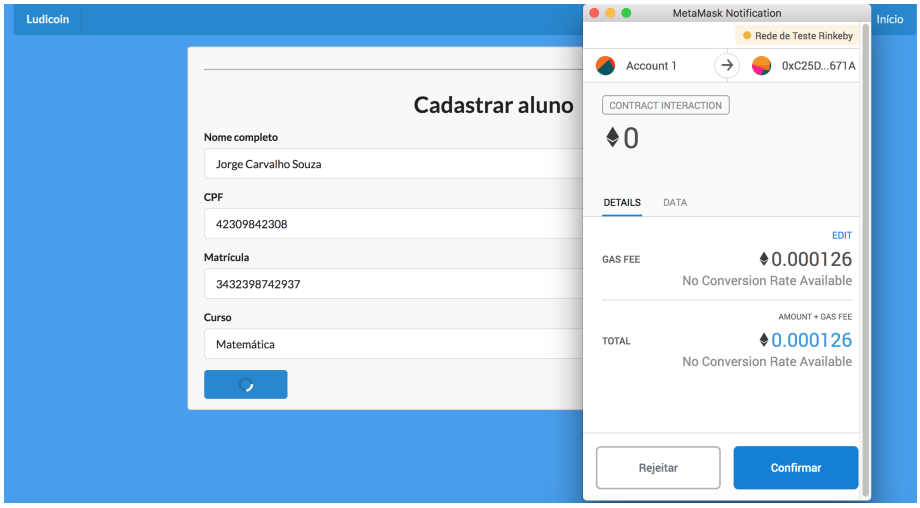
Figura 28 – Exemplo de cartão na página *web*.



Fonte: Próprio autor.

Ao realizar alguma ação que submeta ou altere dados no sistema surgirá um *pop-up* do MetaMask para apresentar o respectivo custo em *ether* e confirmar a transação, ou seja, a ação a ser realizada. O botão clicado apresentará um ícone de carregamento até a ação ser finalizada pelo sistema. A Figura 29 ilustra esse evento.

Figura 29 – Exemplo de submissão de transação.



Fonte: Próprio autor.

A.3.1 Cadastro

Na tela inicial existem as opções de cadastrar sua conta como aluno ou como professor. O seu cadastro ficará vinculado à sua conta Ethereum do MetaMask. Caso se cadastre como professor, o seu cadastro deverá ser aprovado por algum professor já cadastrado no sistema. Neste caso, por padrão, entre em contato com walan.dn@gmail.com para ser aprovado.

A.3.2 Professor

No acesso como professor, você visualizará suas informações e poderá navegar para as seguintes páginas a partir do menu:

- Disciplinas: página com ações de listar, adicionar ou editar disciplinas;
- Turmas: página com ações de listar, adicionar ou editar turmas do professor;
- Professores: página que lista os professores cadastrados e permite a aprovação da solicitação de cadastro de novos professores

Ao entrar em **Turmas > <Turma-Selecionada>** você poderá matricular alunos na turma, passar atividades e, ao clicar no cartão da atividade, atribuir notas.

A.3.3 Aluno

No acesso como aluno, você visualizará suas informações, turmas em que está matriculado e seu saldo em Ludicoins. Nesta página, você terá as seguintes opções:

- Realizar matrícula: você poderá se matricular nas turmas cadastradas pelos professores do sistema. A matrícula precisa ser aprovada pelo professor da turma. Por padrão, entre em contato com walan.dn@gmail.com para ter sua matrícula aprovada.
- Ver notas: apresentará todas as notas de todas as matérias do aluno;
- Minhas turmas: ao clicar no cartão de uma das turmas listadas, você irá para uma página com detalhes sobre a turma, assim como suas notas nas atividades e unidades da turma e poderá gastar Ludicoins.

APÊNDICE B – Comentários e Sugestões do Questionário

Segue reproduzido na íntegra todos os comentários e sugestões deixados no questionário aplicado:

1. "O sistema precisa ser seguro, porque alguns alunos vão buscar explorar o sistema para obter vantagens indevidas."
2. "Usar ludicoins para crédito no resun."
3. "Sobre a pergunta 2, não necessariamente precisa ser pontuação maior que 10 numa unidade de disciplina. Podem ser remunerados ludicoins à atividades específicas, atividades extras, bom desempenho e engajamento nas aulas "não necessariamente relativo à notas", mas sim interação com o professor, resolução de exercícios, proatividade, etc. Acredito que os professores merecem maior flexibilidade para remunerar ludicoins aos discentes. Naturalmente esse sistema econômico-acadêmico se ajustará quanto ao câmbio de ludicoins de uma disciplina pra outra. Um hipotético docente que distribua mts ludicoins acabará com ludicoins desvalorizados em relação à cotação vigente no departamento. É natural que os professores se ajustem e aprendama valorizar seus ludicoins. Outra ideia é a de que professores tenham uma quantidade limitada de ludicoins para distribuir por semestre, logo terá que usá-los com sapiência."
4. "Visto que uma das ideias do projeto é motivar o aluno, seria interessante conseguir ludicoins com base na assiduidade para incentivar os alunos a não faltarem suas aulas."
5. "Padronizar os valores da moeda é mais viável e retira as complicações inerentes as adaptações. Além disso, a moeda não deve ser apenas para alunos que tiram 10, mas também para a realização de atividades, ou também oferecer caráter autônomo, sendo assim aberto para o professor decidir a forma de usá-lo. Além disso, a moeda deve atingir caráter mais geral, permitindo não só adquirir pontuações, mas abrir possibilidade para ganhar descontos em eventos ou ganhar créditos complementares."
6. "Concordo com tudo o que foi exposto e acrescento uma coisa: os alunos da Universidade deveriam receber mais atenção por parte dos docentes, fornecer aulas extras, monitorias e atividades de extensão."
7. "Gerar 'Ludicoins' a partir de atividades extras elaboradas pelo próprio departamento do aluno."

8. "O crédito da matéria talvez deva ser levado em consideração na conversão de ponto para ludcoin. Nota maior que 10 é algo raro, possivelmente tornando a moeda obsoleta pelas poucas situações em que ela poderia ser usada."

9. "Interessante,mas pouco viável."

10. "A pontuação poderia ser equivalente à pontuação de uma determinada unidade ou atividade específica (como projetos). Usando projetos como exemplo, alguns professores costumam dividir a nota em partes ou percentagens da nota, sendo que em alguns casos você acaba com um excesso de ponto em alguma dessas partes, esses pontos também poderiam gerar ludicoins, desde que seja equivalente à pontuação original."

11. "Acredito que esse tipo de sistema pode ser integrado com outros tipos de atividades para além das disciplinas, como projeto de extensão, pesquisa e atividades esportivas."

12. "Seria bom se saísse da teoria e fosse para a prática... Bastante interessante!!"

13. "Num cenário de implementação do LUDICOIN seria interessante poder gerá-los também através de atividades como monitoria e estágios, ou até mesmo em jogos esportivos dos times da UFS, pra estimular a busca dos alunos por todas as atividades da vida acadêmica."

14. "Sinceramente, não faz sentido."

15. "Achei legal. Esse negócio de poder usar sua excelência em outras matérias que você sentiu dificuldade é muito valida. Mas isso deve ser atrelado a pesos, pois principalmente na elétrica a galera ia usar isso para passar em disciplinas "estratégicas"Os pontos podem ter "peso"proporcional a quantidade de créditos da disciplina a fim de representar a importância da disciplina de forma equivalente a grade curricular."

16. "Show."

17. "Eu ficaria mais motivado com esse tipo de troca."

18. "Acho muito interessante. Seria legal ver uma limitação de uso, como um "fundo de emergência"contra reprovação. No geral, acho que o uso dessas Ludicoins para aumentar a média sem uma verdadeira necessidade (por exemplo, o risco de reprovação) pode acabar transformando o boletim do aluno uma grande massagem de ego. Além de transformar o histórico em algo injusto em seleções nacionais e internacionais para mestrados e / ou bolsas. Mas o foda é que pra ser justa, a universidade teria que demitir 70% dos professores cuzões que acham bonitinho reprovar."

19. "Idéia interessante, porém deveria ter a participação de gestores departamentais e de membros da reitoria, pq senão será muito difícil sair do papel e vai acabar sendo mais um projeto com idéia interessante mas irreal/inaplicável."

20. "ludicoins é um nome difícil inventem outro mais ’marketing’."

21. "Penso que um dos pontos mais positivos no qual o Ludicoin pode ser efetivo é na possibilidade de um excelente desempenho em uma disciplina para qual tenho aptidão poder ser transferido em pontos para uma na qual tenho dificuldade."

22. "Discordo completamente do item 2 porque o aluno pode guardar os "ludicoins"para usar em uma disciplina que ele precisa mais, assim sendo aprovado em duas disciplinas ao invés de passar em uma com nota 10 e reprovar em outra."

23. "Acredito que a conversão do ludicoin depende de qual disciplina o aluno ganhou os pontos e qual ele vai utilizar. Por exemplo, um aluno com muitos ludicoins em cálculo 1 pode aproveitá-los relativamente bem cálculo 2, porém não muito em métodos e técnicas de pesquisas."